Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 170 (2020) 584–593

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 6-9, 2020, Warsaw, Poland

Queue based Vehicular Ad Hoc Network Prognostic Offloading Approach

Sony Guntuka[a,*], Elhadi M. Shakshuki[a], Siddardha Kaja[a], Ansar Yasar[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

Vehicular Ad hoc NETworking (VANET) enables a vehicle to connect with other vehicles and the surrounding devices such as Road Side Units (RSUs) and base-stations through a wireless network. There are challenging issues within VANET environment caused by the high demand of Internet access. These issues include an increase in the vehicle traffic and the necessity of dynamic topologies. Nowadays, the high usage of Internet in vehicles is also increasing the load on the cellular network base-stations. To alleviate the load from the base-stations, vehicles should be able to switch the communication between the cellular network and RSUs to offload the data. When a vehicle is not within the RSU signal range, it is still possible for the vehicle to exchange information using Vehicle-to-Vehicle (V2V) communication. The main aim of this paper is to predict the vehicles topology, identify multiple offloading paths and compute the costs of the identified paths. Towards this end, knowledge defined network is utilized. To deal with connection interruptions in V2V, we develop algorithms for predicting an efficient V2V offloading path using queues. These algorithms make it possible to reduce the response time, improve the resource management of the network and helps in efficient service connectivity.

Keywords: VANET; RSU; Offloading path; Knowledge Defined Networking

* Corresponding author. Tel.: +19025851524; fax: +19025851067.
E-mail address: 152874g@acadiau.ca

10.1016/j.procs.2020.03.129

1. Introduction

Vehicular Ad hoc NETworking (VANET) facilitate the communication between vehicles and the Internet. The increasing demand of accessing Internet whenever and wherever is high; thereby, increasing data load on the cellular network base-stations. To reduce the load from the base-stations and satisfy the increasing demand for the Internet, a dedicated short-range communication (DSRC) network is used, which are fixed on the roadside called Road Side Units (RSUs). The deployment of RSUs improve the communication between moving vehicles and the Internet. Typically, vehicles communicate with other entities of the infrastructure through RSUs are called Vehicle-to-Infrastructure (V2I) communication. Vehicles communicate with each other directly using DSRC through On Board Units (OBU) called Vehicle-to-Vehicle (V2V) communication, as shown in Fig. 1. Vehicles continue to communicate through cellular networks when there is no RSU signal.

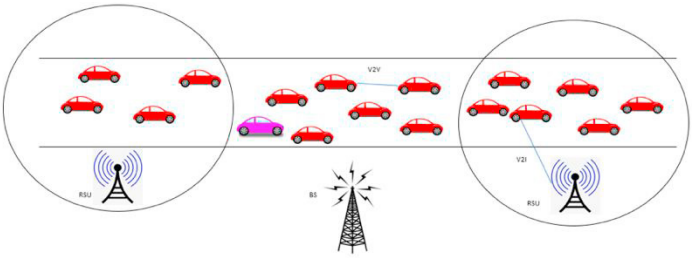

Fig. 1. VANET environment.

VANETs are connected, by default, to cellular network to exchange the data. Whenever the vehicle is in the range of an RSU, it should switch the connection from cellular network to RSU. The typical signal coverage of an RSU is approximately one square kilometer [1]. A vehicle can pass a signal range of RSU in no more than two minutes; if an average vehicle speed of 30 to 40 km/h is assumed. However, there are high chances that a vehicle picks up a high speed and gets out of the signal range of RSU very quickly. Whenever there is no RSU available for the vehicle to communicate, the vehicle is connected to the cellular network and the data is carried by the cellular networks; this in turn increases the data load on base-stations. Offloading reduces the amount of data being carried on the cellular bands, and consequently freeing bandwidth for other users. There are two cases worth mentioning when offloading takes place using RSU. The first case when a vehicle is in the signal range of an RSU. The second case when there exists a V2V path to communicate with an RSU.

1.1 Related Work

Research in the transport network management field and VANET are gaining momentum. Much of this research considering VANET system level design and also node level design [2,3,4].

Vladyko, A. *et.al.* [17] proposed an architecture for assisting the ultra-low latency VANET systems using Software Defined Network (SDN) technology and Mobile Edge Computing (MEC) at Radio Access Network (RAN). Introducing the combination of SDN and MEC in their architecture, the computation is managed and handled at the edge of the network which decreases the network congestion at the core network. Their experimental work proved to achieve better efficiency in terms of packet delivery time and packet loss. For autonomous vehicles, ubiquitous connectivity is a desirable feature. To achieve this, Aljeri, N. and Boukerche, A. [3] proposed an approach that utilizes the concept of neural networks. This neural network predicts the vehicles upcoming Access Router (AR) connection. They claimed that the low computation time and the high prediction accuracy can be achieved. In their approach, probabilistic neural network [2,10] classification model is used to estimate the next most probable AR transition given the vehicles mobile characteristics (such as location, direction, and speed). Huang, C. and Wu, Z. [5] presented the data offloading technique and discussed the VANET architecture using the concept of Mobile Edge Computing (MEC). The key idea of this architecture is to show how a vehicle can offload its data when there is a V2V offloading path with k hops to reach an RSU. The presented scenarios are based on the vehicles position from the RSU. In their work, they discussed how the k-hop V2V offloading path is implemented in these scenarios. As well as, they introduced the parameter of staying time of the offloading vehicle under an RSU signal coverage. In our proposed

approach, we utilized similar staying time parameter. We also utilized the concept of Knowledge Defined Networking (KDN) for implementing the Artificial Neural Network (ANN) through the knowledge plane, similar to the work presented in [7]. The ANN model presented in [7] predicts the network performance for multiple paths identified and showed the experimental results that the increased performance of KDN based data center. A detailed information about KDN is provided in [6], and a detailed information about the concept of Knowledge plane in SDN architecture are provided in [8].

2. The Proposed Approach

This section provides a detailed description of our proposed approach. The road traffic in real time is considered as a Non-Homogenous Poisson Process (NHPP) [1]. The rate of NHPP is a function of time. In order to reduce the complexity of the environment, a unidirectional traffic of vehicles is considered. We also considered the vehicle traffic is homogeneous Poisson process, which is independent of time. To project real time situations, we assume the signal coverage of the deployed RSUs are not overlapping.

The main focus of our approach is to overcome the challenges with the real time data of the vehicles' traffic; specially, the fast-changing topology of the moving vehicles. If we could predict the vehicles position after a certain time, then we would be able to avoid network delays. It should be noted that RSUs are not programmable. To introduce the programmability feature at the network level, we therefore need to utilize the concept of Knowledge Defined Network (KDN). KDN is an advancement of Software Defined Network (SDN) [12,13].

In the SDN architecture, the control and data planes are disaggregated as shown in Fig. 2. The underlying network infrastructure in data plane is abstracted from the applications. Also, the network intelligence and state are logically centralized in the control plane. The controller in the control plane exchanges data with the abstracted application plane and data plane using northbound and southbound interfaces respectively. Knowledge-defined networking is a networking paradigm that allows users to develop artificial intelligence techniques for network operation and control. KDN is dependent on SDN to retrieve the usual telemetry and network analytics and introduces the knowledge plane [14] for network control and management operations.

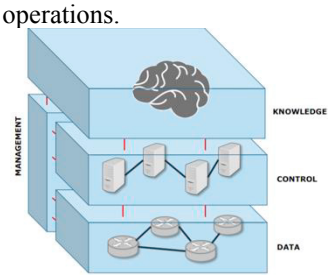

Fig. 2. Knowledge Defined Network planes [6].

The data plane in the KDN architecture provides all the node level network information. This data along with the network analytics are fed to the input layer of the neural network in the knowledge plane. The neural network takes the input data that includes the vehicle parameters received from the Global Positioning System (GPS) [15,16], such as location, direction and the speed of the vehicle. The output of the neural network is a set of V2V offloading paths. The link parameters such as bandwidth, transmission latency, etc., are predicted and then the cost of each path [11] is computed. The cost of the path is determined based on the settings that put forth by the network administrator. We utilize a queue data structure to save all paths with their corresponding costs after they are sorted. In this process, the paths and their costs are enqueued by least cost so that the least cost path is dequeued first. The vehicles in the signal coverage of next RSU are enqueued by staying time [5] of the vehicle so that the vehicle with maximum staying time is dequeued first. The vehicle with the highest staying time under the next RSU is dequeued and used to build a V2V path. The staying time is defined as the time a vehicle is expected to stay in the RSU signal range.

2.1. Path Cost Evaluation

The networks' performance is affected by the path chosen by the network. Various network variables are considered for computing the path's cost [13]. The metrics are varied for different networks, because they depend on

the requirements set by the administrator. To this end, we used a neural network in our proposed approach to find the best V2V offloading path. This is achieved by utilizing queue data structure for the identified V2V paths computed costs. The following parameters are utilized to calculate the networks' performance and used by the neural network for best results.

- *Average Bandwidth* (BW_{AVG}). If one path contains several links ($L_1, L_2, \ldots, L_n$) with the corresponding bandwidth utilization ratio ($BW_1, BW_2, \ldots, BW_n$), the available bandwidth ratio of this path can be calculated by the average of the bandwidths, as follows:

$$BW_{AVG} = \frac{1}{n}\sum_{i=1}^{n} BW_i .$$

- *Hop Count* (NUM_{hops}). The hop count refers to the number of intermediate devices, I, through which data must pass between source and destination. The path with low number of hops is the shortest path and will be the best path to consider ($NUM_{hops} = I\text{-}1$).

- *Transmission Latency* (L). The latency of each link, L_i, can be calculated as the ratio of the transmitted number of bytes, N_{bytes}, to the transmission rate, T_{rate}. The overall latency is the average latencies of all the links.

$$L_i = \frac{N_{bytes}}{T_{rate}},$$
$$L = \frac{1}{n}\sum_{i=1}^{n} L_i .$$

- *Utilization* (U). Network utilization is the ratio of current network traffic, $N_{current}$, to the maximum traffic, N_{max}, that the port can handle. The maximum utilization of the predicted V2V path is the best path.

$$U = \frac{N_{current}}{N_{max}}.$$

- *Throughput* (TP): It is the number of information units that can be processed or moved successfully in a designated amount of time. It can be calculated by use the data size, D, and time, T, as follows:

$$TP = \frac{D}{T}.$$

- *Response Time* (RT): The time interval that begins with accepting a task or a request, T_{accept}, to responding, $T_{respond}$, to a task or a request from the server. This can be calculated as follows and should be minimized for best results.

$$RT = T_{respond} - T_{accept}.$$

- *Energy Consumption* (EC): An efficient and optimal offloading path reduces the energy consumption as this makes better utilization of the resources. Energy consumption can be defined as the required amount of energy for completing a task. The energy consumed by the tasks T_1, T_2,…, T_n is the average of individual energies consumed.

$$EC = \frac{1}{n}\sum_{i=1}^{n} T_i .$$

- *Execution Time* (ET): The time spent by the system to execute a specific task. This is not the same as a response time. In our approach, this is related to computing time to identify multiple V2V paths and calculating the cost of the path and staying time of the vehicle in RSU.

2.2. The Proposed Algorithms

Considering the dynamic behaviour of the vehicle's location, it is expected that it is possible that there are connection interruptions in the predicted V2V. It should be noted that the VANET environment that we deal with is to be predicted after a given time t. In this section, we discuss our proposed algorithms with possible real time scenarios in mind that may occur in a VANET environment. A V2V path is the path that connects the source vehicle

with an RSU through one or more vehicles. The following are some situations that may occur in real time:

- A best-case scenario. In this situation the source vehicle is within the signal coverage of an RSU.
- The source vehicle is not in the signal range of RSU and a V2V offloading path exists.
- The source vehicle is not in the signal range of RSU and the existing V2V offloading path is interrupted. This situation results into two possible cases.
  - Case 1: Path interrupted because of a vehicle not in the RSU signal range.
  - Case 2: Path interrupted because of an offloading vehicle within the signal range of RSU.

This approach uses artificial intelligence techniques through neural networks in the network level, which are programmed using knowledge defined networking. To start with, a data set is given as an input to the neural network, as shown in Fig. 3. For selecting the input, we consider *k*-order Markov model [9], which specifies the events to be considered for predictions. The *k*-order Markov model assumes that the location of the vehicle at time *t* depends on the most recent activity history of the vehicle. Let us consider there are *n* events happened in the history of a single vehicle $H = \{e_1, e_2, e_3, \ldots, e_n\}$. The most recent *k* events are $H_k = \{e_{n-k+1}\ e_{n-k+2}, e_{n-k+3}, \ldots, e_n\}$. A random event e_i belongs to H_k which can be represented as $e_i = \{(x_i, y_i, v_i, t_i, d_i) \mid 1<= i <=k\}$, where *i* belongs to [1,*n*), t_i is the time stamp of the event happened at $t_i < t$, v_i is the vehicle speed, and (x_i,y_i) denotes the coordinates of the vehicle location. Also, assume a set of vehicles $V = \{V_1, V_2, V_3, \ldots, V_m\}$ in a segment S_p that are not in the signal coverage of any RSU and they are using the cellular network to communicate. These vehicles try to connect to the ahead RSU R_{p+1}, where *m* is the total number of vehicles in the segment S_p.

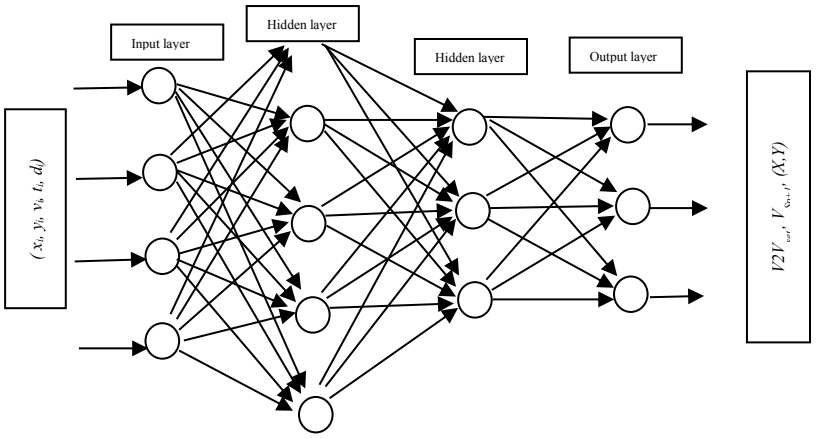

Fig. 3. Basic Neural Network Model.

A sample data set input of the neural network is provided in Table 1. The data set explains the trajectory data generated by GPS by a specific vehicle. Time is represented by t_i, the coordinates of the vehicle (*xi,yi*) by positions *X* and *Y*, three direction by *X, Y, Z*, and the speed of the moving vehicle v_i by speed in Km/h. (*X,Y*) is a set such that each $(X,Y)_j$ are the predicted coordinates for a vehicle $V_j \in V$. In our approach, we assume it is unidirectional traffic. Hence, the trajectory probability model with a single motion pattern [16] can be utilized to predict the location of the vehicle at time *t*.

Table 1. Sample data set.

Time	position X	position Y	direction X	direction Y	direction Z	Speed in Km/h
1.71573	6798.471191	13262.82959	0.51717782	0	-0.855877995	17.57944508
1.98171	6809.912598	13243.89209	0.51706332	0	-0.855947137	21.99577904
2.0317	6798.479492	13262.81641	0.517216325	0	-0.85585469	18.05904293
2.08171	6809.927246	13243.86768	0.51706332	0	-0.855947137	22.56219463
2.13171	6798.488281	13262.80176	0.51726234	0	-0.855826914	18.52701244

Notations:

H – a set of vehicle history events

H_k – set of recent k history events

e_i – a random event in H_k

V – set of vehicles

R_{p+1} – RSU ahead of vehicle V_j

(x_i, y_i) – current coordinates of vehicle V_j

v_i – current speed of vehicle V_j

t_i – time stamp of vehicle V_j at (x_i,y_i)

(X,Y) – a set of predicted coordinates of all the vehicles in V after time t

$(X,Y)_j$ – predicted coordinates of V_j after time t

V_{Sp+1} – a set of vehicles predicted to be in the signal range of R_{p+1}

V_{max} – a vehicle under R_{p+1} having highest staying time

S_T – Staying time

Q_{Rp+1} – Queue of vehicles in set V_{Sp+1} sorted by staying time

$V2V_{set}$ – Set of V2V predicted paths

Q_{path} – Queue of all V2V paths sorted by path cost

$VV_{leastcost}$ – a V2V path with the lowest cost

x – number of V2V predicted paths

$V_{failure}$ – interrupted connection of vehicle V_j with R_{p+1}

V_{repair} – reconnecting vehicle V_j with R_{p+1}

The following describes our proposed algorithms.

Algorithm 1: Vehicle Coordinates Prediction

1. Input: H, V
2. Output: (X,Y)
3. Repeat for each V_j in V // for all the vehicles in V
   - 3.1. Repeat for each H_k in H // for all vehicles' recent history
     - 3.1.1. For each ej in Hk // for each event in the vehicles' history
       - 3.1.1.1. Calculate $(X,Y)j = f(x_i, y_i, v_i, t_i)$ // predict vehicle coordinates
     - 3.1.2. End for
   - 3.2. End for
4. End for

Algorithm 2: Vehicles' staying time

1. Input: (X,Y)
2. Output: V_{max} *in* V_{Sp+1} // V_{max} is a vehicle with maximum staying time
3. While (V_j is in the range of R_{p+1}) Do // check if the vehicle in RSU range
   - 3.1. Create a set V_{Sp+1} // create a set to store vehicles under RSU range
   - 3.2. For each V_j in V_{Sp+1} // for all the vehicles in set V_{Sp+1}
     - 3.2.1. Calculate S_T // compute staying times
   - 3.3. End for

3.4. Create/Refresh Q_{Rp+1} // create a queue to store vehicle information
3.5. For each V_j in V_{Sp+1} // for all the vehicles in set V_{Sp+1}
3.5.1. Enqueue S_T_j in Q_{Rp+1} // insert the vehicle staying time
3.5.2. Sort Q_{Rp+1} from least to highest //sort the queue such that vehicle S_T is maximum
3.6. End for
3.7. Dequeue (Q_{Rp+1}) as V_{max} // get the highest staying time
4. End while

Algorithm 3: Integral V2V Paths

Input: (X,Y), V_{max}
Output: $V2V_{set}$ // a set of possible V2V paths between V_{source} and V_{max}
1. For each V_j in V //repeat this for all the vehicles in the segment
1.1 $V_{source} = V_j$ //trying the paths from the selected vehicle
1.2 $V2V_{set}$ = BFS (V_{source}, V_{max}) //using Breadth First Search for finding the paths
2. End for
3. Create queue Q_{path} *//creating a queue data structure for storing the paths identified*
4. For each $V2V$ in $V2V_{set}$ //repeat this for all the paths in $V2V_{set}$
4.1 Compute path cost //calculate the cost of the paths cost=*f(BW, L, ET, EC, RT)*
4.2 Enqueue (Q_{path}) //insert the path in the queue
5. End for
6. Q_{path} = Sort (Q_{path}) //sorting the paths in the queue based on the cost function values
7. $VV_{leastcost}$ = Dequeue (Q_{path}) //retrieve the path with least cost from the queue

Algorithm 4: Queue based Offloading

Input: Q_{path}, Q_{Rp+1}, $VV_{leastcost}$, V_{max}
Output: *Offload*(*data*)
1. While (Q_{path}<0) Do // for all the V2V paths in the queue
1.1 If ($VV_{leastcost}$) // least path cost
1.1.1 Connect (V_{source} -> R_{p+1}) // connection established between source vehicle and RSU
1.1.2 *Offload (data)* // data offloading
1.2 Else
1.2.1 Find ($V_{failure}$) // find the failure node
1.2.2 If ($V_{failure}$ *equals* V_{max}) // failure node is a vehicle in the signal range of RSU
1.2.2.1 V_{max} = Execute Algorithm 2 // select another vehicle in RSU signal range
1.2.2.2 Repair (V_{source} ->V_{max} -> R_{p+1}) // repair the path
1.2.2.3 *Offload* (*data*) // data offloading
1.2.3 Else // failure node is a vehicle not in RSU signal range
1.2.3.1 If (V_{repair}) // a repair node exists
1.2.3.1.1 Repair (V_{source} ->V_{repair} -> V_{max} -> R_{p+1}) // repair the path
1.2.3.1.2 *Offload* (*data*) // data offloading
1.2.3.2 Else // repair node does not exist
1.2.3.2.1 $VV_{leastcost}$ = Execute Algorithm 3// choose next V2V path with low cost
1.2.3.2.2 Go to Step 2
1.2.3.3 End If
1.2.4 End If
1.3 End If
2 End While

The proposed algorithms are demonstrated and discussed on the following real time scenarios. Consider a vehicle, V_{source}, road segment, S_p, road segment next to S_p is S_{p+1}, RSU in segment S_p is R_p and next RSU at which the vehicle V_{source} is trying to offload is R_{p+1}. V_{max} is the vehicle in the signal coverage of R_{p+1}, which is used in V2V offloading path. Assume we are predicting an offloading path which is about to happen at time t. Fig. 4 shows the environment that we used to demonstrate the scenarios.

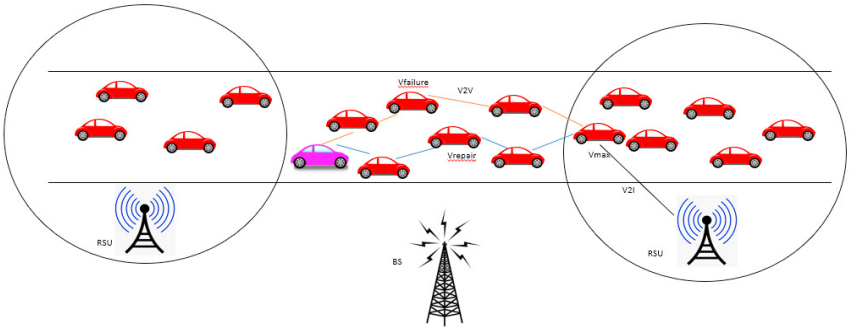

Fig. 4. Test cases scenarios

The best and the worst cases that may occur in real time vehicle traffic are considered and categorized into various scenarios and discussed. The first and straight forward scenario, which we consider as a simple and best-case scenario, is when a vehicle V_{source} in a road segment S_p is in the signal coverage of RSU R_p. Vehicle V_{source} offloads data through R_p. The second scenario that we considered is when the vehicle V_{source} is reaching a point p located in segment S_{p+1} in time t (calculated based on direction, location and speed of the vehicle). Vehicle V_{source} doesn't have a direct path to reach and offload the data to R_{p+1}, because it is not within RSU signal coverage. But, V_{source} can reach R_{p+1} using a *V2V* offloading path.

$$V2V = V_{source} \text{ -> } V_1 \text{ -> } V_2 \text{ -> } V_3 \text{ -> } \ldots \text{ -> } V_{max} \text{ -> } R_{p+1}$$

The third scenario is when the vehicle V_{source} is reaching a point p located in segment S_{p+1} in time t. Vehicle V_{source} doesn't have a direct path to reach and offload the data through R_{p+1}, because V_{source} is not within the RSU signal coverage. But, V_{source} can reach R_{p+1} using a *V2V* offloading path. In this scenario, there are three possible cases that may happen and categorized based on the connection that interrupting the vehicle, as follows:

- The predicted offloading path doesn't work when a vehicle V_j between V_{source} and V_{max} is interrupting the path. A vehicle V_{repair} exists between these two vehicles to repair the path and maintain the communication.
- The predicted offloading path doesn't work when a vehicle V_j between V_{source} and V_{max} is interrupting the path and there is no other vehicle V_{repair} exists between these two vehicles to repair the path. In this case, the least cost path $VV_{leastcost}$ stored in the queue Q_{path} is dequeued as the offloading path.
- The predicted offloading path doesn't work when the vehicle V_{max} is interrupting the path. In this case, the vehicle with maximum staying time under the RSU R_{p+1} which is the next available vehicle in the queue Q_{Rp+1} is to be considered as V_{max} for offloading.

3. Implementation

In this section, we describe the VANET environment used in our proposed approach. To demonstrate the operations of the proposed algorithms presented in this paper, we used Vehicular network simulation (Veins) framework. In the following, we discuss the simulation methodologies, the simulation configuration, and show some results.

3.1. Simulation Methodologies

For simulation purposes, we utilized OMNET++ due to its domain specific nature for supporting wireless ad hoc networks. An open source framework Veins is used for VANET network simulations. The Veins framework makes

the vehicular network simulations as realistic as possible. We are using Veins with a road traffic simulator Simulation of Urban Mobility (SUMO). SUMO is an open source which is designed to handle large road networks. This combination of OMNET++ with Veins and SUMO helps us to demonstrate our proposed algorithms.

3.2. Simulation Configurations

Our simulation is conducted within the OMNET++ 5.5.1 environment with INET framework in a windows 10 operating system. The system is running on a laptop with Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz and 8-GB RAM. To demonstrate our simulation, we have implemented the possible situations in OMNET++ 5.5.1. The default VANET environment configured using Veins with SUMO frameworks includes a set of vehicles, a base station and an RSU.

3.3. Simulation Results

This section provides some results produced from the proposed algorithms, using predefined input data. When our proposed algorithms are utilized in the network, they provided an immediate backup in terms of task and resource management. The algorithm identifies multiple V2V offloading paths to reach an RSU, which reduces the load on the base station. When the selected path is identified to interrupt the connection between the vehicle and the RSU, the next V2V path with least path cost residing in the queue is executed.

The vehicles v8, v9 and v10 shown in Fig. 5. are in the signal range of the RSU. All other vehicles are using the cellular network base station. The data offloading from source vehicle to RSU through wireless networks using V2V communication is demonstrated in the simulations. The resource management and service connectivity of the vehicles in the network are expected to be more efficient when using these proposed algorithms.

Fig. 5. VANET Environment in OMNET++

4. Conclusions and Future work

In this paper, we have introduced the concept of neural networks for predicting a V2V offloading path which is programmed in the Knowledge Plane of Knowledge Defined Network. Several algorithms are proposed for predicting data offloading from a source vehicle to the next available RSU. These algorithms include predicting the coordinates of the vehicles, computing the staying times of vehicles under an RSU, identifying the possible V2V offloading paths, and to handle the connection interruptions. The beacon information of the vehicles such as vehicles location, direction and speed are considered, and the vehicles' position is determined after a certain time frame. This information is used to predict the fast-changing topology of the vehicles in advance. Also, the multiple paths of offloading are predicted. Two queues were introduced one to store all offloading paths that are sorted based on the path cost and the other one to store the vehicles in the signal coverage of the RSU to offload the data that are sorted based on staying time. These queues in the network reduced the computation time when the connection is interrupted while offloading. Our

proposed approach is demonstrated all the possible real time scenarios. The scenarios are also implemented using OMNET++ IDE with SUMO and Veins network simulations. Predicting the offloading path made the resource allocation for the future tasks much easier to handle. Also, a continuous service connectivity is considered and expected when using the proposed approach. In the future, we plan to implement our proposed approach using different neural network techniques and make some comparisons with real collected and distributed data. We also plan to enhance our proposed algorithms with more analyses. We will consider utilizing the combination of different parameters such as bandwidth, latency and other possible parameters that help to predict an efficient path.

References

[1] Yujie Tang, Nan Cheng, Wen Wu and Other (2019) "Delay-Minimization Routing for Heterogeneous VANETs With Machine Learning Based Mobility Prediction" *IEEE Transactions on Vehicular Technology* 68 (4): 3967-3979.

[2] Chen, B.-H., & Huang, S.-C. (2015) "Probabilistic neural networks based moving vehicles extraction algorithm for intelligent traffic surveillance systems" *Information Sciences* 299 :283-295.

[3] Noura Aljeri and Azzedine Boukerche (2019) "A Probabilistic Neural Network-Based Road Side Unit Prediction Scheme for Autonomous Driving" *ICC 2019 - 2019 IEEE International Conference on Communications (ICC*) 1-6.

[4] Sara Mehar, Sidi Mohammed Senouci and Others (2015) "An Optimized Roadside Units (RSU) placement for delay-sensitive applications in vehicular networks" *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)* 121 – 127.

[5] Chung-Ming Huang, Zhong-You Wu (2019) "The Mobile Edge Computing (MEC)-based VANET Data Offloading using the Staying-Time oriented k-hop away Offloading Agent" *2019 International Conference on Information Networking (ICOIN)* 357-362.

[6] Albert Mestres, Alberto Rodriguez-Natal and Others (2017) "Knowledge-Defined Networking" *ACM SIGCOMM Computer Communication Review* 47 (3): 2-10.

[7] Alex M. R. Ruelas, Christian E. Rothenberg (2019) "Load balancing method for KDN-based data center using neural network" *Universidad de Lima* 87-97.

[8] David D. Clark, Craig Partridge, J. Christopher Ramming and John T. Wroclawski (2003) "A Knowledge Plane for the Internet" *SIGCOMM '03 Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* 3-10.

[9] Christine Cheng, Ravi Jain (2003) "Location prediction algorithms for mobile wireless systems" *Wireless internet handbook* 245-263.

[10] Specht, D.F (1990). "Probabilistic neural networks" *Neural networks* 3 (1): 109-118.

[11] U-Chupala, P., Ichikawa, K., Iida, H., Kessaraphong, N., Uthayopas, P., and Others (2014) "Application-Oriented Bandwidth and Latency Aware Routing with Open Flow Network" *2014 IEEE 6th International Conference on Cloud Computing Technology and Science* 775-780.

[12] Open Networking Foundation (2012) "Software-Defined Networking: The New Norm for Networks" *ONF white paper.*

[13] 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.

[14] David D. Clark, Craig Partridge, J. Christopher Ramming and John T. Wroclawski (2003) "A Knowledge Plane for the Internet" *SIGCOMM '03 Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* 3-10.

[15] J. Ghosh, M. J. Beal, H. Q. Ngo, and C. Qiao (2006) "On profiling mobility and predicting locations of wireless users" *Proceeding REALMAN '06 Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality* 55-62.

[16] Lei-lei Wang, Zhi-gang Chen, Jia Wu (2019) "Vehicle trajectory prediction algorithm in vehicular network", *Wireless Networks* 25 (4): 2143–2156.

[17] Andrei Vladyko, Abdukodir Khakimov, Ammar Muthanna, Abdelhamied A. Ateya and Andrey Koucheryavy (2019) "Distributed Edge Computing to Assist Ultra-Low-Latency VANET Applications" *Special Issue Vehicle-to-Everything (V2X) Communication for Intelligent Transportation Systems (ITS)* 11 (6): 128.