Link stability based optimized routing framework for software defined vehicular networks

Kalupahana Liyanage, Kushan Sudheera; Ma, Maode; Chong, Peter Han Joo

2019

Kalupahana Liyanage, K. S., Ma, M., & Chong, P. H. J. (2019). Link stability based optimized routing framework for software defined vehicular networks. IEEE Transactions on Vehicular Technology, 68(3), 2934-2945. doi:10.1109/TVT.2019.2895274

<https://hdl.handle.net/10356/140667>

<https://doi.org/10.1109/TVT.2019.2895274>

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/TVT.2019.2895274>.

Downloaded on 26 Apr 2023 18:08:29 SGT

Link Stability based Optimized Routing Framework for Software Defined Vehicular Networks

Kalupahana Liyanage Kushan Sudheera, *Member, IEEE*

Maode Ma, *Senior Member, IEEE*

Peter Han Joo Chong, *Member, IEEE*

Abstract—The dynamic nature of vehicular networks imposes a lot of challenges in multi-hop data transmission as links are vulnerable in their existence due to associated mobility of vehicles. Thus, packets frequently find it difficult to get through to the destination due to the limited lifetimes of links. The conventional broadcasting based Vehicular Ad-hoc Network (VANET) routing protocols struggle to accurately analyze the link dynamicity due to the unavailability of global information and inefficiencies in their route discovering schemes. However, with the recently emerged Software Defined Vehicular Network (SDVN) paradigm, link stability can be better scrutinized pertaining to the availability of global network information. Thus, in this paper, we introduce an optimization based novel packet routing scheme with a source routing based Flow Instantiation (FI) operation for SDVN. The routing framework closely analyzes the stability of links in selecting the routes and the problem is formulated as a minimum cost capacitated flow problem. Further, an incremental packet allocation scheme is proposed to solve the routing problem in a less time complexity. The objective is to find multiple shortest paths which are collectively stable enough to deliver a given number of packets. The FI scheme efficiently delivers and caches flow information in the required nodes with a reduced extent of communication with the control plane. With the help of realistic simulation, we show that the proposed routing framework excels in terms of performance over the existing routing schemes of both SDVN and conventional VANET.

Index Terms—Optimization, Routing, SDN, VANET

I. INTRODUCTION

Vehicular Ad-hoc Networks (VANET) emerged nearly a decade ago with the primary objective of establishing a proper communication network among vehicles in order to enhance transportation related attributes such as road safety, traffic information dissemination, etc. [1]. Yet, the unique characteristics present in vehicular networks such as high mobility, impose a lot of challenges in realizing these objectives.

The dynamicity associated with vehicular networks causes frequent topology changes. The links are highly vulnerable in their existence and not stable for a considerable time in compared to other ad-hoc networks [2]. As repeatedly emphasized in [3], [4], link dynamics impose serious difficulties in successfully delivering packets. This is crucial especially in

multi-hop data transmission applications such as file sharing, and congestion avoidance, which requires transmission of sequences of packets over multiple hops. As the time frame in multi-hop transmission is fairly large, packets frequently find it hard to reach the destination before the route times out. By the time packets reach an intermediate node, the opening window of the link can disappear leading to unsuccessful transmissions [5]. In fact, studies have found that, in VANET, the route lifetime sharply drops when a path consists of three to four hops and has a high probability of causing a route error [2].

However, distributed VANET routing protocols struggle to cope with this dilemma as they find it difficult to accurately estimate the route validity due to lack of global network information [6]. Hence, the estimated routes may not stably exist until the packets get through, likely resulting in packet drops. This becomes severe, when a considerable time is taken to transmit the packets which could be consequences of a set of packets to be transmitted and/or selection of a long route, as vulnerability increases in both cases. It is highly probable that the packets at the end of the sequence may not be able to reach the destination as a result of a route breach, leading to lower packet delivery ratios (PDR) [5]. Moreover, the system is compelled to find new routes frequently, which could result in higher latencies as well.

The recent induction of Software Defined Networking (SDN) into the vehicular domain has bridged the missing gaps and made global network information available [7]. The resulting Software Defined Vehicular Network (SDVN) provides a bird's eye view over the network, and thus, the link stability can be better scrutinized. Despite having new opportunities to explore various routing attributes such as stability, still, the general focus is on finding in the shortest path [8] and thus, naturally face the problems mentioned above.

Differentiating from the existing studies, in the first part of the paper, we introduce an optimization based novel packet routing protocol for SDVN considering multiple factors, distance and stability. The aim is to select multiple (if requires) shortest paths, which are collectively stable enough to deliver a given number of packets. The overall problem is modeled following a minimum cost capacitated flow problem, and linearized to an Integer Linear Programming (ILP) problem. Further, an incremental packet allocation scheme is introduced as a computational improvement strategy with the help of Bi-directional and K-shortest path algorithms [9].

In SDVN, routing consists of two parts, the routing algorithm and the Flow Instantiation (FI) operation. As the routing information is been generated at the controller, the way this information is delivered to the respective nodes is as equally

Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

Kalupahana Liyanage Kushan Sudheera and Maode Ma are affiliated with School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore. e-mail: (KUSHANSU001@e.ntu.edu.sg, EMDMA@ntu.edu.sg)

Peter Han Joo Chong is affiliated with Department of Electrical and Electronic Engineering at Auckland University of Technology, New Zealand. e-mail: (PETER.CHONG@aut.ac.nz)

Manuscript received XXX, XX, 2017; revised XXX, XX, 2018.

important as the efficiency of the routing algorithm. One of the recent focuses in SDN is on minimizing the extent of communication between the control and data planes, as it disrupts the continuous packet transmission [10], [11], [12]. In SDVN, this is more critical as the controller(s) is(are) generally placed far away from the vehicles [7], [13].

SDN is mainly associated with two FI operations, reactive and proactive [14]. In reactive FI, the controller only installs flow rules in the flow requested node. Therefore, all the nodes along a path that does not possess required flow information are compelled to buffer the data and consult the controller. Hence, the packet flow is frequently disrupted resulting in a high latency and routing overhead¹[13]. To worsen this problem, links in these paths generally have a limited lifetime² due to inherent high mobility [2], and thus, the supplied flow rules will soon expire, creating urges for repeated consultations.

The proactive FI mode avoids the buffer time by actively forwarding required flow rules to all the nodes in the computed path and thus, less disruptive. Even though it helps to reduce the latency, the routing overhead continues to prevail, since almost the same amount of communication exists between the control and data planes. This can lead to an expensive communication cost as well, if a cellular link is used for this interface as in [7]. Further, the proactive FI when used with distributed controllers [15], has struggled to deliver flow rules to all the required nodes, as some vehicle nodes may fall out of local controller's reach, downgrading the performance.

In order to strike a balance between the latency and PDR, and to minimize routing overhead, in the second part of the paper, we present a source routing based FI scheme, which can efficiently deliver flow rules to the respective nodes with a minimum communication with the control plane, and an effective route caching mechanism.

The remainder of the paper is structured as follows. Section II introduces the system model and follows up with a brief literature review on routing in VANET. In Section III and IV, we present our optimization based packet routing algorithm and the FI operation, respectively. Section V analyses the results, and finally, Section VI concludes this paper.

II. BACKGROUND AND RELATED WORK

A. System Model

The main elements in SDVN are the control and data planes, assuming the applications are integrated with control, as shown in Fig. 1. The SDN controller possesses all the logical functionalities and takes decisions, including routing on behalf of the data plane. The data plane consists of vehicle On Board Units (OBUs) and Road Side Units (RSUs), and executes actions specified by the flow rules which are supplied by the control plane [7]. Generally, a data plane node (OBU or RSU) consists of two wireless interfaces, Long Term Evolution (LTE) and Dedicated Short Range Communication (DSRC).

In this paper, we use a hierarchical distributed SDVN (HD-SDVN [15]) architecture as shown in Fig. 1, as the basis, in

Fig. 1. System model (HD-SDVN [15]).

which the control plane is split into two layers, with the top layer on the Internet and the bottom layer at the RSU level. The data plane uses LTE to connect with the top tier, and DSRC to connect with the bottom tier as well as other data plane elements. The adopted architecture generates a collective global network view, while maintaining a lower operational latency as a layer of control is closer to the data plane.

For an accurate computation of routes, a periodic network information collection is required. Differentiating from existing studies, in HD-SDVN [16], we instruct RSUs to gather network information by eavesdropping on the broadcasted basic safety messages (BSMs) in the network. As BSMs include location and velocity information, the RSUs can develop the topology maps based on the estimated distances and the knowledge that they have on the local road network. This network information is first sent to the local controllers, and then to the regional controllers, where they generate aggregated network maps. The network information exchange frequency is set identical to BSM frequency, as only upon arrival of new beacon messages, the network topology changes. Yet, the vehicles that are out of RSU coverage have to periodically update their status in the regional controllers via LTE.

B. Routing in VANET

Conventional VANET routing protocols such as Ad hoc On Demand Distance Vector (AODV) [6], hardly consider link stability in their schemes. However, more recent research such as ROMSGP [3], has focused on minimizing the disruptions caused by the dynamicity of links. In [3], routes are formed within the vehicles in the same velocity group which ensures a more stable selection of links.

Contrarily to the conventional routing approaches where the path is found based on the collective functioning of the nodes, SDVN mostly relies on the control plane for finding the path. The fact that the control plane maintains a global network view has simplified routing in SDVN. Only a few studies can be found in the domain of SDVN routing as per its relative recentness [8], [17], [18], [19]. In [8], the controller estimates the entire path from the source to the destination using the Dijkstra Shortest Path Algorithm (SPA) in response to routing queries. In opposite, the controller partially involves in [17] and guides the nodes to select the next best carriers in a multiple-copy based routing approach. Multiple copies of the data packet are distributed in the network and the controller actively helps to narrow down the propagation of the

¹The routing overhead is defined as the total packet size utilized in delivery of routing information to the required nodes in the respective path(s).

²The term lifetime in fact refers to the residual lifetime of the link.

copies towards the destination. Overall, this routing framework possesses features of both conventional VANET and SDVN routing. Yet, it is apparent that both algorithms neither consider the link lifetime nor the number of packets needed to transfer and hence, viable to the aforementioned problems.

In both above studies, reactive based FI schemes have been used in delivering the flow rules. In [8], route information is only been delivered to the requested node, thus reactive. In the same way, in [17], each hop along the path consults the controller to get information related to the next best carriers. This type of schemes help to achieve a higher PDR as at each node updated route information is been released. Yet, the interaction with the controller at each step highly disrupts the packet movement and generates a high latency and communication cost along with a huge routing overhead.

Alternatively, Liu et al. [18] have adopted a proactive FI scheme to cut down the amount of messages exchanged with the controller. The packets with unassociated flow rules are forwarded to the controller and the potential destinations are evaluated based on the content of packet (ex: accident warning). The controller then actively installs flow rules in all the nodes in the paths and thus, the consecutive packets can reach their destinations without the intervention of the controller. So, the average delay has been significantly reduced, but with a high dependency on the controller.

One recent SDVN study that has considered stability can be found in [19]. However, the algorithm refers to stability as the maximum path duration, which is simply calculated based on the Euclidean distance. The protocol neither considers the wireless channel access constraints nor the mobility associated with links, and primarily focuses on cognitive routing.

III. LINK STABILITY BASED ROUTING ALGORITHM

In this section, we introduce our optimized packet routing protocol, for SDVN which considers multiple attributes as in link stability and shortest travel time in its algorithm. The primary focus is to find multiple paths with lower travel times and with enough stability to deliver a given number of packets.

We adopt a similar approach as in Air Traffic Flow Management (ATFM) where flights are scheduled in the airspace under link capacity constraints [20]. Identically, here we route individual packets between source and destination pairs under link capacity constraints. It should be noted that, we refer the capacity of a link as its ability to deliver packets satisfying a minimum SNR and thus, different from the channel capacity. A link is considered as terminated at the point required receiver sensitivity level (-82 dBm) is failed to achieve.

In compared to ATFM, it is more challenging in vehicular networks due to following reasons.

- 1) Links are short-lived and thus, the capacity of a link is a function of time.
- 2) Wireless channel access imposes limitations on continuous packet transmission.
- 3) Combinational patterns of wired and wireless links make it hard to accurately estimate the packet arrival times.

A. Problem Statement

1) *Background of the Problem:* In the network instance shown in Fig. 2, at a macroscopic level, if the lifetime of

the link 6-7 in the shortest path is 7 ms, and the travel time in a single hop is 5 ms, it is highly likely that the packet will not be able to reach the node 7, as at the time packet reaches node 6 only a 2 ms of opening remaining in the link 6-7, which is not sufficient compared to a travel time of 5 ms.

Moreover, if there are multiple packets to be transmitted, a single path may not be adequate. As per the current example, if the lifetime of the same link, 6-7 is to be 12 ms, only a single packet can be delivered via the path, S-6-7-D and hence, not ideal. A better option would be to transmit the 1st packet via S-6-7-D and the rest of the packets through S-2-3-4-5-D.

Furthermore, a bulk of packets cannot be transmitted consecutively right after the other in a wireless environment, because of the employed channel access mechanisms [21]. After initiating a transmission, the consecutive packets have to wait until the ongoing packet completely reaches the next hop. Even after that if the next consecutive link is wireless, we cannot accurately predict the transmission order as it depends on the random backoff procedure. For an instance, in the network shown in Fig 2, only a single packet transmission can take place at a time in the links of 3-4, 4-5, 5-8 without realizing packet collisions.

The channel access mechanism will not be a concern, if the next link is wired due to different physical characteristics of two channels, and/or if the inter-arrival time of packets is reasonably high. The consecutive links 1-2 and 2-3 shows such a situation. Here, a packet can be transmitted in both links at the same time, as the channels of the two links are different.

2) *Possible Solutions:* If the controller is given information about the number of packets needed to deliver and the gap between the consecutive packets, one option is to select the most stable path as generally followed by research that include stability [3], [19]. Even though this approach may successfully deliver more packets, it might not be effective in terms of latency, as it can lead to longer routes. Given that vehicular network topology is rapidly changing, it is also possible for this approach to not to find a feasible route.

Alternatively, we can always find the shortest path without taking the route stability into consideration. However, as depicted earlier, packets can struggle to reach the destinations with this approach. Though, the shortest path generally implies a lower latency, the fact that the nodes will be compelled to repeatedly consult the controller as the supplied flow rules expire, can limit the latency reduction.

Fig. 2. An instance of a vehicular network.

In our routing scheme, we balance this trade-off between latency and PDR by routing individual packets in the shortest possible paths, which are sufficiently stable in their existence.

In order to realize a low complexity system, we make several assumptions and base rules as listed below.

- Hello and status messages use different frequency bands, and thus, the effect on data transmission is minimal.
- In order to account for the uncertainty affiliated with wireless channel access, we consider the worst case transmission time in links following robust optimization.
- Only existing links in the system are considered in the formulations, as it is not reliable to explore future connections that could establish during the data transmission.

B. Problem Formulation

1) *Notations*: Given a network graph, $G = (V, E)$ where V and E refer to the set of nodes, and edges connecting the nodes, respectively, our aim is to successfully deliver a given number of packets (n) between a source node, s and a destination node, d , with minimum travel time and without violating the dynamic link constraints. Each link from node i to j (refer to as ij) is associated with two metrics. First is the cost metric, C_{ij} , and it indicates the total transmission time of the data packet in link ij . The second metric T_{ij} , refers to the link lifetime at the moment flow rules reach the source.

We use t_{hop}^w and t_{hop}^{wl} to represent the total time to send a packet within a single wired and wireless hop, respectively (hop time). We mainly include the transmission (t_{trans}^x), propagation (t_{prop}^x), queuing (t_q^x) and contention (t_{cont}^x) times in hop time (x represents the link type as in wired or wireless), and neglect the processing time. Detailed expressions of the above terms are included in our previous work [15].

It is important to note that in reality, both t_{hop}^w and t_{hop}^{wl} are affected by other ongoing transmissions. However, as some transmissions do not require the controller's intervention, it is difficult to accurately analyze the effect from ongoing transmissions on the current transmission. Thus, we model t_{hop}^w and t_{hop}^{wl} with added uncertainties as follows.

$$t_{hop}^w = t_{trans}^w + t_{prop}^w + t_q^w + t_{cont}^w + \Delta t^w \quad (1a)$$

$$t_{hop}^{wl} = t_{trans}^{wl} + t_{prop}^{wl} + t_q^{wl} + t_{cont}^{wl} + \Delta t^{wl} \quad (1b)$$

where Δt^w and Δt^{wl} refer to the uncertainty associated with wired and wireless packet transmission, respectively. The upper bounds of (1) are estimated by the controller at each run depending on the parameter values such as packet size.

We use X_{ij} to indicate the packet flow (number of packets) on link ij , and Y_{ij}^p to indicate the p^{th} packet's path through link ij . $H_{s,r}^{p,w}$ and $H_{s,r}^{p,wl}$ represent the number of wired and wireless hops, respectively, from the source node, s to an intermediate node, r via the estimated route for the packet p . $T_{s,r}^p$ refers to the time p^{th} packet takes to reach an intermediate node, r from the source node, s . $N(=|V|)$ and $L(=|E|)$ refer to the number of nodes and links in the network, respectively.

2) *Objective function*: Our objective is to route the individual packets in the vehicular wireless network with the least amount of delay and error rate, thus, the objective function can be formulated as in (2).

$$\text{minimize} \quad \sum_{i=1}^N \sum_{j=1}^N \left\{ C_t \cdot t_{hop}^{ij} \cdot X_{ij} + C_v \cdot V_{eff}^{ij} \cdot X_{ij} \right\} \quad (2)$$

Here, t_{hop}^{ij} will be equivalent to t_{hop}^w or t_{hop}^{wl} depending on the link type. Further, V_{eff}^{ij} refers to the effective velocity associated with the link ij and can be estimated as in (3).

$$V_{eff}^{ij} = \sqrt{V_i^2 + V_j^2} \quad (3)$$

where V_i represents the velocity of node i .

The terms, C_t and C_v refer to the coefficients of total transmission time and effective velocity, respectively, and $C_t \gg C_v$. Hence, our primary focus is on delivering packets with the least travel time which we define as the shortest path. If multiple paths exist with same travel time, the scheme goes for the path with less mobility. As empirically evaluated in [21], at higher effective velocities, the Doppler spread can have a negative impact on error rate. Thus, we reward the selection of less mobile links in the route selection.

3) Constraints:

• Routing

The following equations (4, 5, 6, and 7) are based on the flow conservation concept, and are responsible for routing the n packets in the network. Equation (7) helps to avoid loops in the routing path, pertaining to the bi-directionality of links.

$$\sum_{j \in N_s^{out}} Y_{sj}^p = 1, \quad \forall p \quad (4)$$

$$-\sum_{i \in N_k^{in}} Y_{ik}^p + \sum_{j \in N_k^{out}} Y_{kj}^p = 0, \quad \forall k, p \quad (5)$$

$$-\sum_{i \in N_d^{in}} Y_{id}^p = -1, \quad \forall p \quad (6)$$

$$Y_{ij}^p + Y_{ji}^p \leq 1, \quad \forall p, i, j \quad (7)$$

where N_i^{in} and N_i^{out} refer to the set of neighbors with incoming links to and outgoing links from node i , respectively.

$$X_{ij} = \sum_{p=1}^n Y_{ij}^p, \quad \forall i, j \quad (8)$$

Equation (8) accumulates the number of individual packets in the link ij and equates it to the packet flow of that link.

• Timing

Equation (9a) and (9b) calculate the number of wired and wireless hops, respectively, from the source, s to an intermediate node, r for the packet p . The binary variable Y_{ij}^p will be 1 in the links where packet p travels leading up to the node r .

$$H_{s,r}^{p,w} = \sum_{ij \in L_{s,r}^{u,w}} Y_{ij}^p, \quad \forall p, r \quad (9a)$$

$$H_{s,r}^{p,wl} = \sum_{ij \in L_{s,r}^{u,wl}} Y_{ij}^p, \quad \forall p, r \quad (9b)$$

where $L_{s,r}^{u,w}$ and $L_{s,r}^{u,wl}$ indicate the set of all wired and wireless upstream links of node r from node s , respectively.

The packet p 's arrival time to node r , ($T_{s,r}^p$) primarily depends on the inter-arrival time of packets, t_{gap} . If the t_{gap}

is considerably higher than t_{hop}^{wl} , then the packet transmission will not be disturbed by preceding and succeeding transmissions as once the next packet is due, the previous packet will be far away from the current node's coverage. Thus, we formulate $T_{s,r}^p$ when $t_{gap} \gg t_{hop}^{wl}$ as in (10).

$$T_{s,r}^p = (p-1)t_{gap} + H_{s,r}^{p,w} * t_{hop}^w + H_{s,r}^{p,wl} * t_{hop}^{wl} \quad (10)$$

When t_{gap} is very small, it becomes very complex to accurately model the time a packet takes to reach a given node due to the involvement of randomness. However, adopting robustness, we make the following assumptions to narrow down the possible node patterns to few cases.

- First and last links of the path are wireless. This limits the application of the work to transmissions starting from vehicles and ending in vehicles.
- Two neighboring nodes in a selected path are on the edges of each other's transmission range as the model selects the shortest paths. Thus, two nodes in a computed path that both have a common neighbor will not be neighbors.

Case 1: $H_{s,d}^{p,w} = 0$, $H_{s,d}^{p,wl} = 2$

This is the most basic case in multi-hop data transmission scenario where the source and destination nodes are within 2 hop distance. Here, both hops have to be wireless due to the above assumptions. As the randomness is involved in packet transmission, we assume that the next packet will start to transmit only after the previous packet reaches the destination. Thus, $T_{s,r}^p$ can be expressed as in (11).

$$T_{s,r}^p = H_{s,r}^{p,wl} * t_{hop}^{wl} + 2(p-1).t_{hop}^{wl} \quad (11)$$

Case 2: $H_{s,d}^{p,w} \geq 1$, $H_{s,d}^{p,wl} = 2$

In this case, routing path consists of multiple wired links in between the first and last wireless links. Therefore, the packet transmission will not be disturbed by the preceding and succeeding packets. Hence, the packets can be continuously transmitted with a gap of t_{hop}^{wl} between them.

$$T_{s,r}^p = H_{s,r}^{p,w} * t_{hop}^w + H_{s,r}^{p,wl} * t_{hop}^{wl} + (p-1).t_{hop}^{wl} \quad (12)$$

Case 3: $H_{s,d}^{p,wl} \geq 3$

In cases where there are more than 3 wireless links as shown in Fig. 3, it is difficult to accurately predict the time a packet arrives at an intermediate node as it depends on the orientation of the network. However, in order to achieve robustness, we plan for the worst case which is a minimum of 3 wireless links sequentially located in the path. This is the bottleneck of the path in terms of continuous packet flow.

Due to the employed channel access mechanisms, only a single packet can be transmitted at a time in 3 consecutive wireless links without a possible collision. Yet, the way packets are transferred in these links is unpredictable. But it is safe to say that, there should be a gap of $3t_{hop}^{wl}$ in between consecutive packets in order to avoid collisions at all scenarios.

Fig. 3. Graphical illustration of case 3 scenario.

Fig. 4. Illustration of a collision situation under a packet gap of $2 * t_{hop}^{wl}$.

A packet collision situation with a smaller packet gap is shown in Fig. 4. Therefore, in this case, $T_{s,r}^p$ can be expressed as,

$$T_{s,r}^p = H_{s,r}^{p,w} * t_{hop}^w + H_{s,r}^{p,wl} * t_{hop}^{wl} + 3(p-1).t_{hop}^{wl} \quad (13)$$

• Capacity

The capacity of a link is a function of time, however, in this problem, it is only considered at several discrete points (at packet arrival times), hence, a discrete time problem.

The capacity of link ij at the time packet p reaches node i , U_{ij}^p , can be expressed as in (14).

$$U_{ij}^p = \max(0, \lfloor \frac{T_{ij} - T_{s,i}^p}{t_{hop}^{ij}} \rfloor), \quad \forall i, j, p \quad (14)$$

where t_{hop}^{ij} refers to the packet transmission time in the considered link, ij . Depending on the medium of the link, it will be equal to t_{hop}^w or t_{hop}^{wl} .

Finally, the capacity constraint can be represented as in (15).

$$Y_{ij}^p \leq U_{ij}^p, \quad \forall i, j, p \quad (15)$$

C. Solution Schemes

We can solve the above routing problem globally using a linear optimization solver. However, as the network grows larger, the solution time and complexity also increase. Moreover, in several situations, the network will not be able to find routes which are stable enough to deliver all the packets. In these situations, the optimization solver will result in an infeasible output. Yet, it is possible that there exist some stable routes that can deliver a limited number of packets out of the total packets. In order to retrieve this partial routing information, the optimization problem has to be executed step wisely decrementing the number of packets until a proper result is found. This yields a very high processing time.

Therefore, in this subsection, we introduce an alternative incremental packet allocation scheme which can effectively solve the above routing problem. Later, we compare the computation time of both global and incremental optimization schemes against vehicle density and number of packets.

1) *Global Optimization:* In this subsection, we discuss how to solve the above routing problem as a global optimization problem using a linear solver such as Gurobi [22]. But first, we need to transform several formulations into linear form.

The $T_{s,r}^p$ expressions under channel restrictions (11, 12, 13), and link capacity representation (14) are formulated using logical conditions, and thus, need to be converted into linear form. For that, we use mixed logic constraints [23], which is a mathematical technique used in the translation of logical and nonlinear conditions into linear form.

We introduce three binary variables (b_1 , b_2 , b_3) as follows to assist in generalizing and linearizing the $T_{s,r}^p$ expression.

$$\text{Case 1 : } (H_{s,r}^{p,wl} + H_{s,r}^{p,w} \leq 2) \iff [b_1 = 1] \quad (16a)$$

$$\text{Case 2 : } (H_{s,r}^{p,wl} \leq 2 \wedge H_{s,r}^{p,w} \geq 1) \iff [b_2 = 1] \quad (16b)$$

$$\text{Case 3 : } (H_{s,r}^{p,wl} \geq 3) \iff [b_3 = 1] \quad (16c)$$

The mixed logic constraints can be directly applied to the case 1 and 3 in the following manner.

$$g(\text{case } 1) = H_{s,r}^{p,wl} + H_{s,r}^{p,w} - 2 \quad (17a)$$

$$g(\text{case } 3) = 3 - H_{s,r}^{p,wl} \quad (17b)$$

In general, case i ($i \in \{1, 3\}$) can be linearized as,

$$g(\text{case } i) \leq M_i(1 - b_i) \quad (18a)$$

$$g(\text{case } i) \geq \epsilon + (m_i - \epsilon)b_i \quad (18b)$$

where $M_i (\triangleq \max : g(\text{case } i))$ are sufficiently big and $m_i (\triangleq \min : g(\text{case } i))$ are small integers. ϵ is the machine precision, beyond which the constraint is violated ($\epsilon > 0, \epsilon \sim 0$).

For the case 2, we introduce another two auxiliary binary variables as b_2^1, b_2^2 such that $b_2 = b_2^1.b_2^2$ and,

$$(H_{s,r}^{p,w} \geq 1) \iff [b_2^1 = 1] \quad (19a)$$

$$(H_{s,r}^{p,wl} \leq 2) \iff [b_2^2 = 1] \quad (19b)$$

The non-linear AND logic expression, ($b_2 = b_2^1.b_2^2$) requires further linearization as follows.

$$-b_2^1 + b_2 \leq 0 \quad (20a)$$

$$-b_2^2 + b_2 \leq 0 \quad (20b)$$

$$b_2^1 + b_2^2 - b_2 \leq 1 \quad (20c)$$

The final representation of $T_{s,r}^p$ is formulated as in (21).

$$T_{s,r}^p = H_{s,r}^{p,wl}.t_{hop}^{wl} + H_{s,r}^{p,w}.t_{hop}^{w} + 2(p-1).t_{hop}^{wl}.b_1 + (p-1).t_{hop}^{wl}.b_2 + 3(p-1).t_{hop}^{wl}.b_3 \quad (21)$$

In addition to timing constraints, the link capacity representation (14) also requires to be linearized. In order to achieve that, we introduce an auxiliary variable, $\gamma(r, p)$, as follows.

$$f(r, p) = \frac{T_{ij} - T_{s,r}^p}{t_{hop}^{ij}} \quad (22)$$

$$[f(r, p) \geq 0] \iff [\gamma(r, p) = 1] \quad (23a)$$

$$[f(r, p) < 0] \iff [\gamma(r, p) = 0] \quad (23b)$$

Linearization of the above logical equivalences follows the same format as in (16, 17, 18). However, to save space, detailed mathematical expressions are excluded from the paper. Nevertheless, it leads the formulations to $U_{ij}^p = \lfloor f(i, p) \rfloor \cdot \gamma(i, p)$. Here, the floor function can be mathematically formulated as follows.

$$U_{ij}^p \leq \frac{T_{ij}}{t_{hop}^{ij}} \cdot \gamma(i, p) - \frac{1}{t_{hop}^{ij}} \cdot T_{s,i}^p \cdot \gamma(i, p) \quad (24a)$$

$$U_{ij}^p > \frac{T_{ij}}{t_{hop}^{ij}} \cdot \gamma(i, p) - \frac{1}{t_{hop}^{ij}} \cdot T_{s,i}^p \cdot \gamma(i, p) - \gamma(i, p) \quad (24b)$$

Still, there exists a nonlinear term in the formulations, ($\tau(i, p) = T_{s,i}^p \cdot \gamma(i, p)$), and needs to be linearized using mixed logic constraints, similar to (20). The final linear representation of U_{ij}^p is shown in (25).

$$U_{ij}^p \leq \frac{T_{ij}}{t_{hop}^{ij}} \cdot \gamma(i, p) - \frac{1}{t_{hop}^{ij}} \cdot \tau(i, p) \quad (25a)$$

$$U_{ij}^p > \frac{T_{ij}}{t_{hop}^{ij}} \cdot \gamma(i, p) - \frac{1}{t_{hop}^{ij}} \cdot \tau(i, p) - \gamma(i, p) \quad (25b)$$

Overall formulation of the above packet routing problem adopts features from maximum flow problem and minimum cost flow problem. Moreover, as all the formulations including the objective function and constraints are linearized, the overall problem is an Integer Linear Programming problem (ILP).

2) *Incremental Optimization*: In this subsection, we introduce the proposed incremental packet allocation scheme which can solve the previously modeled routing problem in a less time complexity [24]. The incremental packet allocation scheme is shown in pseudo-code in Algorithm 1. The basic motivation behind this scheme is to allocate packets as much as possible to the consecutive shortest paths without exceeding the lifetime based link capacities.

Algorithm 1 Incremental Packet Allocation Scheme

Inputs: P, S, D $\triangleright$ P-Number of packets, S - Source node, D - Destination node

Output: results

- 1: (cP, n, LSI) $\leftarrow$ 0; $\triangleright$ cP - current packets, n - explored paths, LSI - last success path index
 - 2: **while** $cP < P \wedge n < C$ **do**
 - 3: $path[n] \leftarrow \text{BiDirectionalDijkstra}(S, D)$;
 - 4: $ij \leftarrow \text{findBottleneckLink}(path[n])$;
 - 5: **if** $\text{feasibilityCheck}(path[n], ij, S)$ **then**
 - 6: $p' \leftarrow cP$;
 - 7: **while** $\frac{T_{ij} - T_{s,i}^{p'}}{t_{hop}^{ij}} > 1$ **do**
 - 8: $p' \leftarrow +$;
 - 9: $result \leftarrow (p' - cP, path[n])$;
 - 10: $cP \leftarrow p'$;
 - 11: $LSI \leftarrow n$;
 - 12: $n \leftarrow +$;
 - 13: **if** ($cP < P$) **then**
 - 14: $result \leftarrow (P - cP, path[LSI])$;
 - 15: **return** result;
-

As illustrated in the algorithm, we iteratively find the n^{th} shortest path with n incremented at each step. A combination of Bi-directional Dijkstra SPA and K-shortest path algorithm [9] is used to find successive shortest paths. Moreover, we use $(C_t \cdot t_{hop}^{ij} + C_v \cdot V_{eff}^{ij})$ as the cost metric to empower both lower latency and error rate in route selection.

As links in a vehicular environment are bidirectional, we can employ the bidirectional SPA. In this algorithm, rather than exploring nodes only from the source node, we start to explore from both source and destination nodes. The node exploring continues until both circles meet in a middle node, and the shortest path is found by extracting the minimum distance from different nodes in two circles. With respect to the conventional Dijkstra SPA, the bi-directional SPA explores lesser number of nodes, and thus, faster.

In order to find successive paths, we use a modified version of the Yen's SPA [9]. Here, we first find the shortest path

using the bi-directional SPA and then in order to find the subsequent shortest paths, we delete individual links stepwise in the previously found path and find shortest paths from the resulting networks. Thereafter, the path with the least distance value will be selected as the next best shortest path.

Next, before assessing the stability of the path, we extract out the bottleneck link of the path, as it is a qualified representation of the path's stability. It can be found out by estimating the remaining link lifetimes at the times considered packet reaches the starting node of the links, and finding the link with the minimum value as formulated in (26). If the bottleneck link satisfies the previously modeled link capacity constraints, then it can be concluded that all the links in the path successfully satisfy the link capacity constraint. This yields a competitive edge over the global optimization in which all the links are assessed by the link capacity constraints, and further, one of the main reasons for the higher computational efficiency.

$$\text{link } ij \text{ index} = \frac{T_{ij} - T_{s,r}^p}{t_{hop}^{ij}} \quad (26a)$$

$$\text{bottleneck link index} = \min : (\text{link } ij \text{ index}) \quad (26b)$$

where p = number of currently allocated packets + 1.

If the selected path passes the feasibility test, we allocate packets as much as possible to the corresponding route until the link capacity is not exceeded as shown by the lines 7, 8 in Algorithm 1. Then, if there are still packets to be allocated, the algorithm goes for a higher shortest path by incrementing n . The process continues until either all the packets are allocated or maximum permissible paths are explored. If the maximum permissible path limit is reached, the remaining packets will be allocated to the last successfully explored path, as it is the most stable path with the highest probability of successfully transmitting the packets. With respect to the global optimization scheme, the incremental scheme is capable of finding a finite set of packet allocated routes in a lesser time.

IV. SOURCE ROUTING BASED FI SCHEME

In general, for a FI operation to be effective, the extent of communication between the control and data planes, and the time spent on delivering the flow rules should be minimized while maximizing the number of flow rules successfully installed in the respective nodes. To achieve these goals, we introduce an effective FI scheme based on source routing and selective route caching, as follows.

In SDVN, when a node receives a packet, it first checks for a matching entry in its flow table. If a match is found, the corresponding action will be performed. If not, a *packet in* message which includes information about the source and destination nodes, is generated and forwarded to the controller. In response to a *packet in* message, the control plane computes the required route using the previously described routing algorithm. The contribution of this section lies in the way this routing information is delivered to the respective nodes.

A. Route Valid Time Estimation

Before sending the flow rules, the control plane will estimate whether the flow information should be stored at intermediate

nodes, based on the valid time of the route. The valid time of a route is different at different nodes in the path. At a considered node, the valid time is equal to the minimum link lifetime of its all downstream links in the path as expressed in (27).

$$\text{Route valid time at node } r, \quad T_r^v = \min.(t_{ij}) \quad (27)$$

where $ij \in L_r^D$

where L_r^D refers to the set of downstream links in the path from node r and t_{ij} represents the lifetime of link ij .

The lifetime of a link primarily depends on the current position of the two nodes that formed the link, and their relative velocities. It can be modeled as a simple sub-optimization problem as in (28) with Fig. 5 as the reference [13].

maximize: (Δt_{ij})

subject to $\{x_i - x_j + [v_i^x(t_o) - v_j^x(t_o)]\Delta t_{i,j} +$

$$\frac{a_i^x - a_j^x}{2} \Delta t_{i,j}^2\}^2 + \{y_i - y_j + [v_i^y(t_o) - v_j^y(t_o)]\Delta t_{i,j} + \frac{a_i^y - a_j^y}{2} \Delta t_{i,j}^2\}^2 \leq D_R^2(\bar{x}, \bar{y}) \quad (28)$$

where Δt_{ij} refers to the residual lifetime of link ij , t_o indicates the time of the last received update, and x_i, x_j and y_i, y_j represent the x and y coordinates of the two nodes. $D_R(\bar{x}, \bar{y})$ indicates the statistically estimated transmission range of the nodes at $\bar{x}$ and $\bar{y}$, in which $\bar{x} = \frac{x_i + x_j}{2}$, $\bar{y} = \frac{y_i + y_j}{2}$.

The effective transmission range is affected by the obstacles in that particular area. Thus, the use of a constant value for the maximum transmission range, may lead to incorrect estimations of link lifetime. To mitigate this effect, we statistically profile the maximum effective transmission range with respect to the position and generate Probability Density Functions (PDFs), and contour maps.

The terms, a_i^x and a_i^y refer to the x and y components of node i 's acceleration. $v_i^x(t)$, $v_i^y(t)$ represent the projected x and y velocity components of node i at time t . Here, the vehicle speed is not treated as a deterministic constant, and its variable nature is acknowledged. Differentiating from the existing studies, which model the speed as a stochastic variable with a probabilistic approach [25], we exploit the knowledge

Fig. 5. Illustration of vehicle interconnection and velocity profiles.

(a) Candidate vehicle selection

(b) Vehicle velocity projection

Fig. 6. Selection of candidate vehicles and projection of vehicle velocity.

that the control plane has over the vehicles' recent speed profiles. Thus, our approach is to predict the vehicle's speed profile in near future using curve fitting (regression) models. Existing literature shows that a considered vehicle's speed is affected by the speed of vehicles in-front [25]. Thus, the set of vehicles that is interest to a considered vehicle's speed will be selected based on the location information and velocity vectors as shown in Fig. 6a. From the vehicles within the given boundary, those who show positive dot products in terms of placement ($\vec{V}_1 \cdot \vec{AB} > 0$) and velocity ($\vec{V}_1 \cdot \vec{V}_2 > 0$) with the considered vehicle will be selected as vehicles with interest.

We use both linear and nonlinear regression models in predicting the velocity of the vehicle. The fundamental concept is to select the function with the least sum of error. The sum of error can be represented as in (29).

$$\text{Sum of error} = \sum_{i=1}^n C_i [Y_i - f(t_i)]^2 \quad (29)$$

where n refers to the number of data points, $f(t)$ is the predicted velocity profile, Y_i is the recorded velocity of the considered vehicle at time t_i , and C_i is the weight of the i^{th} data point. C_i 's value depends on the vehicle that the corresponding data point belongs to. If it is from the considered vehicle, its value will be high. C_i will be low if it is from a vehicle away from the considered vehicle.

In linear regression, $f(t) = a.t + b$, while in nonlinear regression, only a 2nd order polynomial function, $f(t) = a_0 + a_1.t + a_2.t^2$, is considered. The sum of error expression, which is to be minimized is partially differentiated with respect to different coefficients, and solved for the minimum. After getting the best-fitted curve and line, the corresponding sum of errors are calculated and the function with the least sum of error will be selected as the velocity profile of the considered vehicle. An example case is shown in Fig. 6b.

The above problem in (28) can be simply solved by converting the inequality constraint into an equality and then solving the resulting 4th order polynomial equation shown in (30).

$$\begin{aligned} & \Delta t_{ij}^4 [A_{ij}^x{}^2 + A_{ij}^y{}^2] / 4 + \Delta t_{ij}^3 [V_{ij}^x(t_o) \cdot A_{ij}^x + V_{ij}^y(t_o) \cdot A_{ij}^y] / 2 + \\ & \Delta t_{ij}^2 [V_{ij}^x(t_o)^2 + V_{ij}^y(t_o)^2 + (X_{ij} \cdot A_{ij}^x + Y_{ij} \cdot A_{ij}^y) / 2] + \\ & \Delta t_{ij} [2 \cdot X_{ij} V_{ij}^x(t_o) + 2 \cdot Y_{ij} V_{ij}^y(t_o)] + \\ & X_{ij} + Y_{ij} - D_R^2(\bar{x}, \bar{y}) = 0 \end{aligned} \quad (30)$$

where $x_i - x_j = X_{ij}$, $y_i - y_j = Y_{ij}$, $v_i^x(t_o) - v_j^x(t_o) = V_{ij}^x(t_o)$, $v_i^y(t_o) - v_j^y(t_o) = V_{ij}^y(t_o)$, $a_i^x - a_j^x = A_{ij}^x$, and $a_i^y - a_j^y = A_{ij}^y$.

We use Ferrari's method [26] to solve the quartic equation in (30). Its usage is shown in Algorithm 2, in which estimation of route valid time at intermediate nodes is illustrated.

The route caching decision is taken by the controller based on the evaluation of route valid times. If the route shows an enough valid time at a particular node ($>$ data transmission time from the considered node to the destination + time to deliver the flow rules to that node), the decision is taken to cache the route information. The motivation is to save the flow rules only at the nodes which are associated with sufficient route valid times to transmit at least one more packet. Overall, it cuts down unnecessary route caching and helps to lower the communication delay with less work at garbage collection.

Algorithm 2 Computation of Path (Route) Valid Time

Input: $n[N]$, $p[N, N]$, $q[N, N]$, $r[N, N]$, $s[N, N]$, $u[N, N]$
 $\triangleright n[N]$ - set of nodes in the computed path. p, q, r, s, and u are the coefficients of 4th to 1st order terms, respectively, of the quartic equation in (30).
Output: $T_V[N]$ $\triangleright$ Path valid times at each node
1: $T_{low} \leftarrow M$; $\triangleright M$ - very large positive value
2: $T[i, i+1] \leftarrow 0$; $\triangleright$ Link lifetime array
3: **for** $(i = (N-1) \rightarrow 1)$ **do**
4: **if** $(p == 0 \wedge q == 0 \wedge r == 0 \wedge s == 0)$ **then**
5: $T[i, i+1] \leftarrow T_c$; $\triangleright$ Wired: High constant time
6: **else**
7: $T[i, i+1] \leftarrow \text{FerrariMethod}(p[i, i+1], q[i, i+1], r[i, i+1], s[i, i+1], u[i, i+1])$; $\triangleright$ Wireless: Solution of the quartic equation (30).
8: **if** $(T_{low} > T[i, i+1])$ **then**
9: $T_V[i] \leftarrow T[i, i+1]$;
10: $T_{low} \leftarrow T[i, i+1]$;
11: **else**
12: $T_V[i] \leftarrow T_{low}$;
13: **return** T_V ;

B. Source Routing

The computed routing information is then delivered to the respective nodes using source routing. All the path information will be embedded in a modified flow mod message (*flow mod**) and send to the route requested node (usually the source node).

Upon receiving the *flow mod** message which includes sequences of information of intermediate nodes and valid times, the source node will extract the information related to the first path and embeds in the buffered packet and forward to the next hop. The remaining path information (if any) will be used at the designated packet due times.

When the path appended data packet (data*) is transversed in the network, all the intermediate nodes will check the corresponding route valid times in the header and if it is positive, the respective IP address and route timeout, which we refer to as a *segment*, will be stored in their flow tables. Moreover, they will remove the path information related to the visited nodes and shift the header before forwarding to the next hop, and thus, this mechanism generates a lower average routing overhead as graphically depicted in Fig. 7.

C. OpenFlow Realization

In order to make this process a realization, we introduce several modifications to the existing OpenFlow protocol. The

Fig. 7. Proposed flow instantiation operation.

current OpenFlow protocol will update the flow table and send the packet to the next hop without any alteration in response to a *flow mod* [27]. However, we modified this protocol by adding new action types such that upon receiving a *flow mod**, the node will first extract the information related to the first route and then, the first segment of the extracted route will be saved in its flow table and the remaining segments will be appended in the option field of the corresponding buffered data packet. Next, the modified data packet (*data**) will be forwarded in the data plane. The remaining routing information (if any) will be temporarily buffered to process in due times.

Moreover, we introduce new protocol ids in the protocol field of the IPv4 packet format so that a *data** packet can be detected by the OpenFlow protocol. If a match is found for a *data** packet, the corresponding action is to extract the first segment of the routing information from the option field, which reflects the next hop information, and save it in its flow table, similar to *flow mod** operation (refer Fig. 7).

In case of a route breakdown, the respective node will consult the assigned controller for route information and the above process will be re-initiated.

D. Path Encoding

In this subsection, we discuss the packet formatting related to the aforementioned operation. We specify routing as the *reason* in the *packet in* header and embed the number of packets required to be routed in the *data* field, and thus, no alteration to the *packet in* format is required.

As the computed paths will be appended to the *flow mod** message, the classic *flow mod* format will be altered to *flow mod** as shown in Fig. 8. We no longer require the *idle timeout*, *out port* and *out group* fields in the *flow mod* message. Rather, we have a common *hard timeout* and a sequence of segments in the *flow mod** message. Additionally, we include an attribute called *number of paths* in the *flow mod** header, as these messages can carry multiple path information.

As vehicular networks do not have differentiable port numbers [12], we are compelled to use IP addresses to indicate next hop information. Yet, it can be justified by the fact that communication in VANET is usually within few hops in compared to wired networks.

V. PERFORMANCE EVALUATION

In this section, we comparatively evaluate the performance of our proposed routing framework in terms of effectiveness of the results (time complexity and accuracy) and simulation metrics such as packet reception rate, latency and overheads.

A. Agility and Accuracy

First, we compare the time complexity and the accuracy of the proposed incremental optimization based routing algorithm against the global optimization scheme and basic Dijkstra SPA.

Fig. 8. Packet format of *flow mod** message.

Fig. 9. Effectiveness of different solution approaches.

We considered different instances of vehicular networks with varying vehicle density and number of packets and executed the above mentioned three algorithms against multiple source-destination pairs. The considered network map is a simplified version of the Nanyang Technological University (NTU), Singapore campus map with 20 RSUs covering the entire area. The number of nodes was changed from 30 to 100 in steps of 10 in both Figs. 9 and the number of packets was changed from 1 to 16 in Fig. 9a and was fixed at 16 in Fig. 9b.

The Gurobi commercial optimizer [22] was used to solve the global linear interpretation of the routing problem and C++ language was used to implement and solve both incremental optimization scheme and Dijkstra SPA. A workstation with a Intel-core i7 3.40 GHz processor, and a 8 GB RAM was used in performing the experiments. The corresponding results are shown in Fig. 9. The computational time of the Global optimization scheme spans from around 220 ms to 100 s. To avoid anomaly, it is represented as a plane at the top.

It is important to note that both Dijkstra SPA and incremental optimization scheme display processing delays in scale of milliseconds, while global optimization goes even up to few hundreds of seconds. Such a high latency profile is not applicable in vehicular networks as the latency requirement for multi-hop data transmission is generally under 500 ms [28]. The proposed incremental optimization scheme is well within the required limits and thus, satisfactorily meets the requirement. Even though the shortest path approach exhibits a very low processing delay, inconsideration of number packets and stability of routes have adversely affected the packet reception. Meanwhile, the proposed computational strategy almost line up with the global strategy in terms of number of dropped packets and objective value.

B. Simulation Scenarios

In this section, we simulate our optimized routing framework in NS-3 and measure its performance against an existing SDVN routing protocol and conventional VANET routing protocols such as AODV and ROMSGP [3]. We use PDR, average end to end delay, routing overhead and state information acquiring overhead (total network state information acquired by the control plane) as the performance metrics.

1) *Simulation Configuration*: We use the same simplified NTU, Singapore campus map with 20 scattered RSUs as the simulation environment which covers an approximate area of $1500 \times 1500 \text{ m}^2$. Realistic mobility traces are extracted from SUMO for each simulation scenario and fed into NS-3. Hello and beacon message interval is set to 250 ms. Data

TABLE I
SIMULATION PARAMETERS.

Parameter	Value
Simulation time	500 seconds per run with 25 runs per each point
Packet sizes	Packet In = 32 bytes, Flow Mod = 56+ bytes, Data = 1500 bytes
Transmission	Type = UDP, Range = 200 m (DSRC)
Propagation model	Friis Propagation Loss model
Wireless network	DSRC (IEEE 802.11p) = 6 Mbps
Wired network (RSU-RSU)	Ethernet (IEEE 802.3) = 100 Mbps, Propagation delay = 1 ms
Wired network (Controller-RSU)	Broadband = 1 Gbps, Propagation delay = 10 ms
Internet backbone	Data Rate = 10 Gbps, Propagation delay = 10 ms

transmission follows a Poisson distribution with a mean of 4 sec. Each transmission is associated with multiple packets which are randomly selected based on a uniform distribution between 1-10. The packet arrival interval is set to 100 ms. Additional parameters and values are tabulated in Table I.

In order to make the simulations more realistic, we impose a random processing delay for computing the route information at the control plane. For that, we first generated the best-fitted processing delay graphs using our sample data at each number of packets and nodes (Fig. 9a). All the best-fitted graphs resulted in Gaussian distributions. Thus, processing times were randomly selected using corresponding Gaussian distributions.

The overall simulation results are taken by varying the number of vehicles (vehicle maximum speed fixed at 20 ms^{-1}) and vehicle speed (number of vehicles fixed at 60). The results are shown in Fig. 10, 11, 12a and 12b for PDR, average end to end delay, routing overhead and state information collection overhead, respectively. The Fig. 13a and 13b depict theoretical comparisons of FI schemes. Packet overhead and latency overhead respectively refer to the additional packet size and delay consumed in delivering flows.

2) *Results Analysis*: According to the PDR comparison shown in Fig. 10, as expected, the proposed optimized routing scheme displays the highest packet reception ratio at all vehicle density levels and speeds. AODV and shortest path approach have struggled to maintain competitive PDRs due to the inconsideration of stability. Since the ROMSGP utilizes the most stable path, it has maintained a competitive packet reception rate against increasing vehicles and mobility. Yet, inefficiencies in distributed routing schemes and unavailability of global network information have limited its PDR rate. Overall, the proposed routing framework has a competitive edge with a 15–20% PDR gap over the shortest path approach in both graphs (Fig. 10a, 10b). In depth analysis over stability of routes in the proposed framework can be stated as the root cause for the accomplished PDR improvement.

In terms of latency (Fig. 11), AODV clearly suffers a very high delay profile due to their broadcast approach in finding routes. The latency performance gets poorer with increasing number of vehicles and speed, as in both cases, more Route Request (RREQ) packets are broadcasted in the network, making the network more congested. Even though, ROMSGP utilizes a broadcasting approach in finding routes, the higher

Fig. 10. PDR comparison.

Fig. 11. Latency comparison.

Fig. 12. Experimental comparison of routing attributes.

Fig. 13. Theoretical comparison of FI attributes.

stability in selected routes causes less route discovering cycles, and it has helped to exhibits a lower competitive latency. The proposed routing framework presents very low delay patterns against both number of vehicles and speed, mainly owing to the source routing based FI scheme, as it minimizes the extent of communication with the control plane (Fig. 13a). Accurate consideration of link dynamics in the routing algorithm is also equally important in achieving the lower delay profile, as it selects multiple routes beforehand which are stable enough to transmit all the given packets. However, in the shortest path approach, the nodes only receive single path information and are compelled to consult the controller frequently when the route expires, leading to slightly higher communication delays as verified by Fig. 11a and 11b.

This is further evident in the experimental routing overhead

shown in Fig. 12a. According to the graph, the shortest path scheme generates a considerable routing overhead when the vehicle speed increases as the links become vulnerable in their existence with high mobility leading to shorter route valid times. Thus, the nodes are compelled to consult the controller frequently. In the case of ROMSGP, its broadcast based route discovery process is the main cause for the extremely high overhead. However, the proposed routing framework displays a significantly low routing overhead due to its source routing based FI scheme and consideration over link dynamics, as both of them help to minimize the controller consultation.

As shown in Fig. 13a, the proposed FI mode yields the lowest forwarding state distribution against increasing number of hops. The underlying reason is that the controller sends all the path information to the data plane in a single transmission. However, in other FIs, packet overhead linearly increases with higher slopes against the number of hops, as the controller separately delivers flows to each node. Even though, in 13b, the proactive FI displays a very low constant latency overhead, architectural challenges limit its adaptability in SDVN.

The yielded performance improvement is further helped by the offloading of state information collection to RSUs, as it takes out the periodical status message transmission of the vehicles, as empirically shown in Fig. 12b. Overall, the results have comparatively validated the effectiveness of the proposed routing framework and its practicality in the vehicular domain.

VI. CONCLUSION

In this paper, we proposed a novel routing framework for SDVN scrutinizing the stability of routes in addition to the shortest travel time. The routing problem is modeled as an ILP problem. We further introduced an incremental packet allocation scheme to solve the routing problem in a less time complexity. The resulting framework routes the packets via multiple routes along the shortest possible paths with enough stability. Additionally, a source routing based FI operation was introduced to efficiently deliver the routing information to the respective nodes. The proposed FI scheme encapsulates all the routing information in a modified flow mod message and sends only to the route requested node, minimizing the extent of communication between the control and data planes. Based on the route valid times, the path information is only cached at several selected nodes to avoid unnecessary route caching. Via simulation results, we showed that the proposed routing framework improves the performance of multi-hop data transmission in vehicular networks in compared to both existing VANET and SDVN routing protocols.

ACKNOWLEDGMENT

This research is financially supported by the National Research Foundation (NRF) of Singapore (Award No. : NRF2018-SR2001-005).

REFERENCES

[1] F. Dressler, H. Hartenstein, O. Altintas, and O. K. Tonguz, "Inter-vehicle communication: Quo vadis," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 170–177, 2014.

[2] J. J. Blum, A. Eskandarian, and L. J. Hoffman, "Challenges of inter-vehicle ad hoc networks," *Trans. Intell. Transport. Sys.*, vol. 5, no. 4, pp. 347–351, 2004.

[3] T. Taleb, E. Sakhaee, A. Jamalipour, K. Hashimoto, N. Kato, and Y. Nemoto, "A Stable Routing Protocol to Support ITS Services in VANET Networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3337–3347, Nov 2007.

[4] K. L. K. Sudheer, M. Ma, and P. H. J. Chong, "Link Dynamics Based Packet Routing Framework for Software Defined Vehicular Networks," in *2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[5] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *Int. J. Netw. Manag.*, vol. 11, no. 1, pp. 3–30, 2001.

[6] C. Perkins, E. Belding-Royer, and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC Editor, 2003.

[7] I. Ku, L. You, M. Gerla, F. Ongaro, R. L. Gomes, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," in *Ad Hoc Networking Workshop (MED-HOC-NET)*, 2014 13th Annual Mediterranean, 2014, pp. 103–110.

[8] M. Zhu, J. Cao, D. Pang, Z. He, and M. Xu, "SDN-Based Routing for Efficient Message Propagation in VANET," in *Wireless Algorithms, Systems, and Applications: 10th International Conference, WASA 2015, Qufu, China, August 10-12, 2015, Proceedings*, K. Xu and H. Zhu, Eds. Cham: Springer International Publishing, 2015, pp. 788–797.

[9] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[10] H. Li, M. Dong, and K. Ota, "Control Plane Optimization in Software-Defined Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7895–7904, Oct 2016.

[11] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, Aug. 2010.

[12] A. Hari, T. V. Lakshman, and G. Wilfong, "Path Switching: Reduced-state Flow Handling in SDN Using Path Information," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15. ACM, 2015, pp. 36:1–36:7.

[13] K. S. K. L., M. Ma, and P. H. J. Chong, "Efficient flow instantiation via source routing in software defined vehicular networks," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sept 2017, pp. 1–5.

[14] L. MacVittie, "Reactive, Proactive, Predictive: SDN Models," 2013.

[15] K. L. K. Sudheera, M. Ma, G. G. M. N. Ali, and P. H. J. Chong, "Delay efficient software defined networking based architecture for vehicular networks," in *2016 IEEE International Conference on Communication Systems (ICCS)*, 2016, pp. 1–6.

[16] K. S. K. Liyanage, M. Ma, and P. H. J. Chong, "Controller placement optimization in hierarchical distributed software defined vehicular networks," *Computer Networks*, vol. 135, pp. 226 – 239, 2018.

[17] Z. Ming, C. Zhiping, C. Jiannong, and X. Ming, "Efficient multiple-copy routing in software-defined vehicular networks," in *2015 International Conference on Information and Communications Technologies (ICT 2015)*, 2015, pp. 1–6.

[18] L. Yu-Chun, C. Chien, and S. Chakraborty, "A Software Defined Network architecture for GeoBroadcast in VANETs," in *Communications (ICC)*, 2015 IEEE International Conference on, 2015, pp. 6559–6564.

[19] H. Ghafoor and I. Koo, "CR-SDVN: A Cognitive Routing Protocol for Software-Defined Vehicular Networks," *IEEE Sensors Journal*, vol. 18, no. 4, pp. 1761–1772, Feb 2018.

[20] G. G. N. Sandamali, R. Su, Y. Zhang, and Q. Li, "Flight routing and scheduling with departure uncertainties in air traffic flow management," in *2017 13th IEEE International Conference on Control Automation (ICCA)*, July 2017, pp. 301–306.

[21] F. Bai, D. D. Stancil, and H. Krishnan, "Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 329–340.

[22] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: <http://www.gurobi.com>

[23] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[24] A. M. Sharp, "Incremental algorithms: solving problems in a changing world," Ph.D. dissertation, Cornell University, 2007.

[25] M. H. Eiza, Q. Ni, T. Owens, and G. Min, "Investigation of routing reliability of vehicular ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 179, Jul 2013.

[26] G. Cardano, *The Rules of Algebra: (Ars Magna)*, 2007.

- [27] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," *IEEE Communications Surveys & Tutorials*, vol. 17, pp. 27–51, 2015.
- [28] Z. Hameed Mir and F. Filali, "LTE and IEEE 802.11p for vehicular networking: a performance evaluation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–15, 2014.

Kalupahana Liyanage Kushan Sudheera received his B.Sc. Eng. (First Class Honors) degree from University of Ruhuna, Sri Lanka in 2015. He is currently a PhD candidate at Nanyang Technological University, Singapore. He was previously a Lecturer in the Department of Electrical and Information Engineering, Faculty of Engineering, University of Ruhuna, Sri Lanka. His research interests mainly include Software Defined Networking, Vehicular Ad-hoc Networks, Internet of Things/Vehicles, Mobile/Wireless Networks, Optimization methods, Software Engineering.

Dr. Maode Ma received his Ph.D. degree in computer science from Hong Kong University of Science and Technology in 1999. Now, Dr. Ma is an Associate Professor in the School of Electrical and Electronic Engineering at Nanyang Technological University in Singapore. He has extensive research interests including network security and wireless networking. Dr. Ma has more than 380 international academic publications including more than 180 journal papers and about 200 conference papers. He serves as a Senior Editor or an Associate Editor for

other 4 international academic journals. Dr. Ma is a Fellow of IET, a Senior Member of IEEE Communication Society and IEEE Education Society, and a Member of ACM.

Dr. Peter Han Joo Chong is currently a Professor and Head of Department of Electrical and Electronic Engineering at Auckland University of Technology, Auckland, New Zealand. He received the B.Eng. (with distinction) in Electrical Engineering from the Technical University of Nova Scotia, Halifax, NS, Canada, in 1993, and the M.A.Sc. and Ph.D. degrees in Electrical and Computer Engineering from the University of British Columbia, Vancouver, BC, Canada, in 1996 and 2000, respectively. His research interests are in the areas

of mobile communications systems including radio resource management, multiple access, MANETs/VANETs, multihop cellular networks and Internet of Things/Vehicles.