P^{3}R: Realizing Robust Routing for VANET using Trajectory Prediction and Crossroad Recognition

Chuwen Zhang*, Huichen Dai*, Yong Feng*, Yang Li*, Wenquan Xu*,
Xuefeng Ji*, Ying Wan*, Gong Zhang†, Bin Liu*
*Department of Computer Science and Technology, Tsinghua University
†Theory lab, huawei

***Abstract*—High topology dynamics and intermittent connectivity in Vehicular Ad hoc Network (VANET) bring huge challenges to end-to-end communication. Existing routing protocols for MANET such as AODV and OLSR work fine under modest mobility, but have a difficult time to handle frequent topology changes in VANET. This paper proposes Peeking at the Past and Present Routing (P^{3}R), a routing protocol that will calculate next-hops when the past forwarding is considered invalid. The next-hop calculation is based on the predicted locations of forwarder's neighbors and the packet's destination node, overcoming the inaccuracy caused by stale location information. Furthermore, we differentiate vehicles on crossroads as they have high connectivity in actual urban streets. In this way, P^{3}R is able to deal with link breakages quickly and exploit new links. Simulation results show that P^{3}R outperforms state-of-the-art alternatives in terms of packet delivery ratio, delay and cost, while maintaining strong scalability and robustness. We also implement P^{3}R in a real vehicular testbed and the results reveal it has high connectivity on real streets.**

***Index Terms*—VANET, trajectory prediction, crossroad recognition, routing**

I. Introduction

Mobile Ad hoc Network (MANET) is a network comprised entirely of wireless stations. Vehicular Ad hoc Network (VANET) is a specific kind of MANET, where the nodes are vehicles running on the city roads. With the development of Internet of Things (IoT), VANET is now attracting tremendous attention from both the research community and industry for its wide application in the future. E.g., VANET can be used to transmit important or urgent messages (e.g., real-time traffic on roads or traffic accidents) among vehicles without the support of infrastructures; meanwhile, the communication between vehicles can facilitate the autopilot techniques. Therefore, efficient data transmission is the utmost important issue for VANET.

Routing protocols play a vital role in efficient data transmission. Existing routing protocols for MANET fall into two categories: topology-based (e.g. AODV [1] and OLSR [2]) and position-based protocol (GPSR [3]). However, the characteristics of VANET make it difficult to apply the existing routing protocols into it. The nodes in VANET usually have high (can reach 80 km/h) and various velocities on urban streets, and the differences of velocities lead to constant fluxes in the topology and numerous intermittent links. Protocols like OLSR and AODV will be overwhelmed when handling these situations: OLSR will generate torrents of link status change messages, and always be in the convergence stage; AODV is desperately searching for new routes for pending packets; GPSR blindly selects the neighbor closest to the destination without considering real streets and car movements, which leads to many packets being forwarded to a dead end. Therefore, transport layer protocols will experience high loss ratios, long delays, and even timeouts, making MANET not suitable for real-time applications.

Meanwhile, VANET on real streets also brings some opportunities. As vehicle trajectories are restricted by the urban streets, their locations after a short time are predictable. Hence, some schemes [4]–[7] use digital map information or trajectory prediction to route smartly. However, these schemes relying on complex computation and high-level assistant information is too heavy and not suitable for real-time communication.

Faced with this circumstance, we propose the Peeking at Past and Present Routing (P^{3}R), a light-weight and scalable routing protocol based on node locations, exploiting trajectory prediction and crossroad recognition at the same time without assistance from any digital map. The rationale behind P^{3}R is simple: the next hop of a packet is determined by the locations of the current node and the final destination. Each node in P^{3}R periodically sends out *beacon* messages carrying its location information to their immediate (one-hop) neighbors. It also receives location information from their neighbors and maintains such information in the *neighbor list*. Beacon messages are the only protocol messages in P^{3}R. Besides, each node maintains a forwarding table to record the past forwarding strategies. Packets will be forwarded according to the matched table entry unless it is invalid. In this case, the next hop will be selected considering the predicted present location of each neighbor, including its distance and deviation angle to the destination, and if it is at a crossroad.

We simulate P^{3}R on NS-3 and compare it with AODV and GPSR. The results reveal that P^{3}R outperforms AODV and GPSR in terms of delay, cost, and delivery ratio. P^{3}R also shows good scalability when the number or velocity of network nodes increases. We also implement P^{3}R in our dedicated VANET testbed to verify its performance on real streets where wireless channels can be blocked by buildings. Especially, we made the following contributions:

978-1-7281-2583-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICPADS47876.2019.00103

1) We propose a VANET routing protocol called P^{3}R that utilizes trajectory prediction and crossroad recognition to calculate next hops. P^{3}R also keeps historical forwarding strategies as ancillary information.
2) P^{3}R scales better than existing routing protocols because it only keeps the location information of immediate neighbors and historical forwarding information.
3) We simulate P^{3}R, and the results indicate P^{3}R is superior to AODV and GPSR in many aspects. We implement P^{3}R on our VANET testbed, and the experiments prove its high performance on real streets.

The rest of the paper is organized as follows. Section II surveys the related work, Section III describes the protocol design of P^{3}R, Section IV conducts simulations and compares results with the existing representative ones, and Section V describes the VANET tested and implementation results. At last, we make a conclusion in Section VI.

II. Related Work

Routing protocols in MANET are mainly classified in two different categories, topology-based routing protocol and position-based one. The former can further be divided into proactive routing protocol (e.g., DSDV [8] and OLSR), reactive routing protocol (e.g., DSR [9] and AODV) and hybrid Routing (e.g., ZRP [10]). The most significant advantage of topology-based routing protocol is that they can ensure a complete path from the source to destination, i.e. the packet delivery is smooth while the path keeps working. However, overhead on maintaining the path cannot be ignored. Position-based routing protocol, such as GPSR, GPCR [11] and DREAM [12], makes forwarding decision under GPS information of other nodes. It doesn't need a path from source to destination and is delivered hop by hop. On the other hand, hop-by-hop forwarding under a local optimal principle cannot guarantee the arrival at the destination when nodes are distributed on real streets.

VANET is a particular case of MANET, and the biggest difference between them is that nodes in VANET follow some paved roads instead of moving arbitrarily, which inspires researchers to use the digital map and position prediction to help route. Zhao *et al.* in [4] proposes several VADD protocols to forward the packet to the best road with low data-delivery delay. Naumov *et al.* in [13] presents routing the packets in a greedy manner toward the destination through a set of anchor points and uses the guard node to track the current position of the destination. Similarly, RBVT [5] creates road-based paths consisting of successions of road intersections and packets are forwarded in a greedy manner between intersections on the path, reducing the path's sensitivity to individual node movements. An algorithm adaptively to predict the future positions of mobile nodes using historical records is proposed in [14]. In [15], Namboodiri *et al.* design a prediction-based routing protocol to predict route lifetimes so that it can create new routes before existing ones fail. In [7], Xue *et al.* first extracts vehicular mobility pattern from real trace data, then utilizes the pattern and digital map to assist routing. More information assistance, e.g., the digital map, and complicated prediction method, can bring more accurate results but larger overhead on calculation and bandwidth. So, we care more about a light-weight and map-independent routing protocol under a certain accurate ratio.

VANET is born with the mutual adversary of highly dynamic topology, intermittent connection, and interference between nodes, which has brought great challenges in designing efficient routing protocols. Many protocols based on Delay-Tolerant Network (DTN) [16] can realize high performance on throughput and bandwidth utilization except for delay. E.g., Mobieyes [17] is an efficient application for delivering sensed data, such as license plates and road condition, by using a low cost distributed index to represent the storage of sensed data. Our aim is to design a routing protocol to support delay-sensitive applications, but to the best of our knowledge, protocols above are not suitable. P^{3}R is such a routing protocol for delay-sensitive applications. It works like AODV unless the next hop obtained from the forwarding table is thought to be invalid. At that time, P^{3}R will trigger a calculating process to get the best next hop as GPSR and GPCR does, and then add the calculating result to the forwarding table. In addition, to improve the accuracy of calculating next hop, we propose a smarter algorithm of next hop calculation considering the real roads and node location prediction. No need to run time-consuming recovering process, fully utilizing delivering information in history and calculating next hop with an effective algorithm make P^{3}R perform better on cost, delay, and packet delivery ratio.

III. Protocol Design

This section describes the design of P^{3}R in detail. As aforementioned, P^{3}R uses the locations of nodes, as well as trajectory prediction and crossroad recognition, to calculate next hops. Today it is common that vehicles are equipped with GPS devices, from which the location information can be fetched. Each node in the network shares its location and velocity information to its immediate neighbors via beacon messages, i.e., P^{3}R requires the propagation of topology information for only a single hop. With the location and velocity information, the node can envision its neighbors' future locations, such as whether there exists a neighbor at a crossroad or deviating from the direction to the destination. Exploiting such information can help make a correct forwarding decision, without any other topological information.

The locations of the two end points in an end-to-end communication are carried in the data packet headers. For the first packet to a destination, the location of the destination is unknown to the source, so this packet will be flooded until the destination is reached. The subsequent packets will be able to use the locations of both ends.

While forwarding a packet, if there is no matched entry in the routing table or the matched entry is thought to be invalid, next hop calculation is triggered. And the next hop is computed in this way: 1) predict the current locations of the neighbors based on the locations and velocities conveyed by

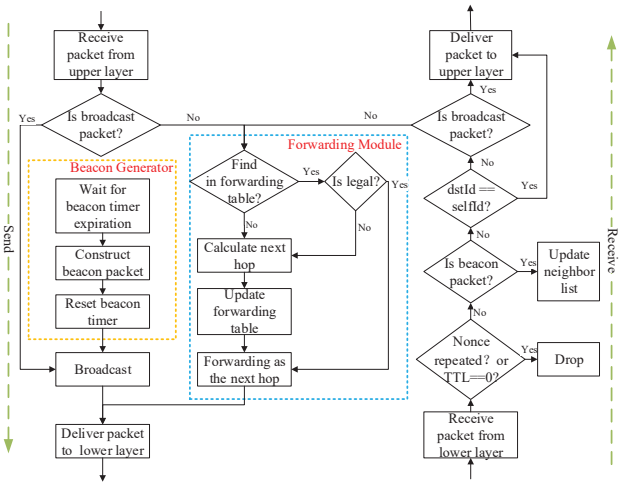


Fig. 1: The flow chart of packet processing in P^{3}R

the beacon messages in recent several rounds; 2) use predicted locations to obtain comprehensive scores to approximate the neighbors' probabilities of reaching the destination and select the neighbor with the highest score. On the other hand, if the matched entry is valid, the packet is forwarded to it.

P^{3}R's packet processing is shown in Fig. 1, including packet sending, receiving, and forwarding. We will explain them below.

A. Location sharing

Node locations, as well as the velocities, need to be shared among nodes by P^{3}R. There are two ways to share such information in P^{3}R: 1) each node periodically sends out beacon messages; 2) the location information is carried by data packets in end-to-end communication.

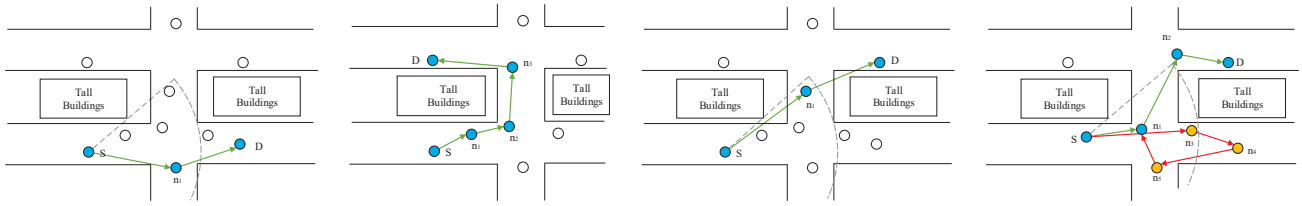
Beacon message contains the ID, velocity, and position of its sender. In order to utilize the broadcast nature of wireless radio communication, beacon messages are broadcast to the neighbors so that it needs to be transmitted only once. Beacon messages are the only protocol messages in P^{3}R, so the cost of P^{3}R is kept at a low level. The beacon message generator is included in the packet sending procedure in Fig. 1. A node receives beacon messages from all of its neighbors and keeps tracking their states in a neighbor list. Each neighbor entry consists of four fields: node ID, location queue, velocity queue, and timestamp, meaning that node specified by the ID has recent locations and velocities in the queues, and the last beacon message is received at the moment indicated by the timestamp. Once a node receives a beacon message, it extracts the location, velocity, and timestamp, and then inserts/updates them to the neighbor list. A neighbor node will be removed from the neighbor list if no beacon message is received from it for the timeout duration. Hence, the neighbor list only stores recent local topologies,

The first packet to a destination may have no idea where the destination node is located, so this packet needs to be flooded until it reaches the destination or a node that has a forwarding table entry to the destination. Indeed, we can obtain the destination location by some schemes based on digital map, such as HCBLS [18]. However, P^{3}R is a dedicated light-weight routing protocol which can works in the darkness, so we keep the flooding process for the first packet if without assistance from digital maps. P^{3}R sets TTL in each packet header to prevent infinite transmission, and a unique identifier in the header called *nonce* to prevent multiple duplicate broadcasts. Both the packet TTL and nonce will be examined as shown in the packet receiving procedure in Fig. 1. While two nodes are communicating, their latest locations are encapsulated in the packet headers. Meanwhile, the locations can also be learned along the end-to-end path. Note that we don't need to predict the destination position as our protocol works on real-time scenarios for delay-sensitive applications where round trip time is on the order of milliseconds, much less than beacon cycle, so the destination movement is small. Besides, as the packets get closer to the destination, they are more likely to obtain a fresher location of the destination from vehicles on the path, which further reduces the error.

B. Calculating next hop

In P^{3}R, the concept of next hop calculation is similar to that of GPCR and GPSR, but it considers the real vehicle distribution and road geometry with intelligence. GPSR always selects the neighbor closest to the destination blindly, which is likely to lead packets lost in the wrong direction. On the contrary, GPCR selects the neighbor located on crossroads with the priority and routes packets as GPSR between crossroads, which is too conservative to cost more hops sometimes. To overcome the drawbacks of them, P^{3}R combines their concepts. We take some examples to illustrate how P^{3}R works in Fig. 2. Assuming the roads are straight and crossed, the forwarding node S should send packets to a proper next hop to reach the destination D. We can divide the problem into three typical cases based on the angle ϕ between the current road direction and the vector $\overrightarrow{SD}$, i.e., parallel ($\phi \approx 0$), vertical ($\phi \approx \pi/2$) and intersecting with an acute angle ($0 < \phi < \pi/2$).

In case one, as shown in Fig. 2(a), the best selection is n_1, which is nearest to D and on the same road as S. As for case two that the forwarding node has no neighbor closer than itself, P^{3}R adopts the right hand rule as shown in Fig. 2(b). The forwarding node S traverses the neighbors clockwise from the vector $\overrightarrow{SD}$ and returns the first encountered node n_1. Packets will be forwarded one by one according to this principle until they reach node n_3. Since D is in the neighborhood of n_3, n_3 sends it directly to D. In case three, if the forwarding node has a neighbor near to the destination and these three nodes are in a straight line, selecting this neighbor is the best choice obviously, as shown in Fig. 2(c). Otherwise, it is properer to select nodes in crossroads to sustain accessibility. As shown in Fig. 2(d), if the forwarding node S selects the node n_3 according to the nearest-to-destination principle, the packet will be forwarded to n_4 and then n_5 as the right hand principle due to the blockage of tall buildings. On the contrary, if S select the node n_1 which is at the crossroad but not the nearest one to the destination, the extra three hops (n_3, n_4 and n_5)



(a) Case 1: send the packet to the neighbor nearest to D directly.

(b) Case 2: use right hand principle to forward the packet.

(c) An example of case 3: there exists a neighbor near the line S to D, and sent the packet to it.

(d) An example of case 3: sending to a node in a crossroad saves more hops than the one nearest to D.

Fig. 2: Three typical cases of vehicle distributions

can be saved. This case is common in cities, so choosing the next hop in crossroads conservatively is wiser sometimes.

Faced with the three typical vehicle distributions above, P^{3}R uses two different modes to calculate next hops: perimeter and greedy mode. The former corresponding to case two, uses the right hand principle to obtain the next hop. The latter containing case one and three, selects the next hop by calculating the direction angle denoted by θ_i (defined as the angle between vectors of forwarding node to destination and neighbor node), distance to destination denoted by d_i and non-crossroad indicator denoted by I_i of each neighbor i, where $I_i = 0$ if the node i is in crossroads, and 1 otherwise.

Before explaining the algorithm of calculating next hop in detail, we first need to solve the problem of calculating θ_i, d_i and I_i. As mentioned above, the neighbor list can provide location and velocity information of neighbors, which can be used for calculating θ_i and d_i. However, the time interval granularity of beacon packets, such as several seconds, is too coarse for high-speed cars in reality, so we should make use of available information in neighbor list and realize accurate trajectory prediction for real cars. P^{3}R adopts the prediction method in [6] as the equation (1) below, which predicts the trajectory under the fact that acceleration denoted by a rather than speed obeys the normal distribution and applies linear regression to predict changes in speed direction (i.e., gradient) denoted by k,

$$\begin{cases} \Delta x_i = v_i \Delta t + \frac{1}{2} a \Delta t^2 \\ k_i = \beta_0 + \beta_1 k_{i1} + \cdots + \beta_m k_{im} \end{cases} \tag{1}$$

where $a \sim N(\mu, \sigma^2)$, v_i represents the newest speed of neighbor i stored in the neighbor list, $(k_{i1}, k_{i2}, \cdots, k_{im})$ represents the continuous m speed directions of neighbor i.

Although the digital map can help to recognize the crossroads effectively, it is not easy and necessary to use digital map application in network layer. Therefore, we prefer to explore the behavior features of cars to recognize crossroads in the neighborhood. Apparently, cars often turn at crossroads and their trajectories also break the straight line, so we propose our crossroad recognition algorithm based on the linear correlation analysis, which uses the location information over the past several time points as the following equation,

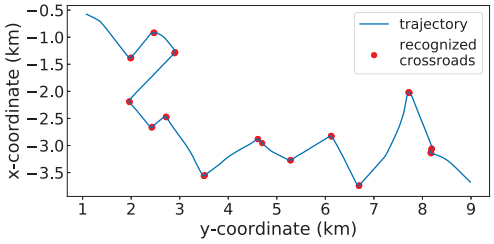


Fig. 3: An example of recognized crossroads on a real car trajectory: 13 rights and 1 mistake.

$$\rho_i = \left| \frac{\sum_{k=1}^{K} (x_{ik} - \overline{x_i})(y_{ik} - \overline{y_i})}{\sqrt{\left(\sum_{k=1}^{K} (x_{ik} - \overline{x_i})^2\right)\left(\sum_{k=1}^{K} (y_{ik} - \overline{y_i})^2\right)}} \right| \tag{2}$$

where ρ_i is the correlation coefficient, which indicates the linearly significant level; K denotes the number of historical locations used; x_{ik} and y_{ik}, the last k-th x-coordinate and y-coordinate of neighbor i; $\overline{x_i}$ and $\overline{y_i}$, the average x-coordinate and y-coordinate of last K locations of neighbor i. We infer the neighbor is at a crossroad, i.e., $I_i = 0$, only if $\rho > 0.9$. Then we collect real car traffic traces and do the linear correlation analysis to determine the minimum of parameter K. The results show that $K = 5$ is accurate enough to recognize crossroads as shown in Fig. 3.

Therefore, P^{3}R uses the history locations of its neighbors to determine whether there are crossroads in its neighborhood, and maintains a crossroad table to store their locations. If we recognize any location of a turning car as a crossroad, one real crossroad might be mistaken as several ones. To solve this mistake, a new crossroad table entry is inserted unless it is not close (e.g., more than 20 m) to any other existing crossroad. If it is recognized as a duplicate crossroad, P^{3}R will update the crossroad location by the average of these two locations.

So far, with key parameters of neighbors, d_i, θ_i and I_i, P^{3}R is supposed to make a next hop choice based on these information to realize high packet delivery ratio, low delay and cost. Apparently, if a neighbor has lowest d_i, θ_i and it is also at a crossroad, it is definitely the best decision from local view, but this case is not common. Hence, this

decision problem can be transformed into a multi-objective optimization problem, and we can unify these parameters through linear combination as follows. 1) Normalize these three parameters. P^{3}R uses feature scaling method to normalize d_i, i.e., $\widetilde{d}_i = d_i - \min(d_i)/(\max(d_i) - \min(d_i))$. As θ_i is in the range of 0 to π, P^{3}R uses $\widetilde{\theta}_i = \theta_i/\pi$ instead. As I_i is within 0 to 1 naturally, it keeps unchanged. 2) Traverse the neighbor list to solve the optimization problems below.

$$\begin{aligned} \min_{i} \quad & \alpha_1 \widetilde{d}_i + \alpha_2 \widetilde{\theta}_i + \alpha_3 I_i \\ s.t. \quad & \alpha_1 + \alpha_2 + \alpha_3 = 1, \quad i = 1, 2, \cdots, N \end{aligned} \tag{3}$$

where $\alpha_1, \alpha_2, \alpha_3$ are weight coefficients, which can be set by expertise or optimized through technical methods, such as machine learning, and this is not involved in P^{3}R temporarily.

Accordingly, the algorithm of next hop calculation in P^{3}R is listed in Algorithm 1. First, if the neighbor list is empty, the algorithm will return false immediately as the first two lines show. Otherwise, the forwarding node will traverse its neighbor list as line 6 to 16, in which it updates the node information by trajectory prediction in equation (1), calculates new d, θ and I, and adds all the new nodes to set C. In addition, it obtains the maximum and minimum distance of all neighbors to the destination. Then, it uses the right hand principle if the minimum distance is less than the distance from the forwarding node to destination (17 to 18). Otherwise, it traverses the set C to get the score of each neighbor. Finally, it returns the neighbor with the minimum score (line 20 to 24).

Algorithm 1 CNH: Calculating Next Hop

Input: Destination node dst, forwarding node $self$
Output: Next hop node $nextHop$

1: **if** $self.neighbors == \emptyset$ **then**
2: **return** $false$
3: $C \leftarrow \emptyset$
4: $disMin \leftarrow +\infty, disMin \leftarrow 0, scoreMin = 1$
5: **for** $node$ in $self.neibhbor$ **do**
6: $\Delta t \leftarrow$ GETTIME() $- node.timestamp$
7: $nodeNew.pos \leftarrow$ LOCPREDICT$(node, \Delta t)$
8: $nodeNew.d \leftarrow$ DISTANCE$(dst, nodeNew)$
9: $nodeNew.\theta \leftarrow$ ANGLE$(dst, nodeNew)$
10: use $nodeNew.d$ to update $disMin$ and $disMax$
11: $nodeNew.I = 1$
12: **for** j in crossroads **do**
13: **if** DISTANCE$(nodeNew, j) <$ RADIUS **then**
14: $nodeNew.I = 0$
15: **break**
16: add $nodeNew$ to C
17: **if** $disMin \geq$ DISTANCE$(dst, self)$ **then**
18: use right hand principle to get $nextHop$
19: **else**
20: **for** $node$ in C **do**
21: $score \leftarrow \alpha_1 \frac{node.d - disMin}{disMax - disMin} + \alpha_2 \frac{node.\theta}{\pi} + \alpha_3 \cdot node.I$
22: **if** $score \leq scoreMin$ **then**
23: $nextHop \leftarrow node$
24: **return** $nextHop$

C. Maintain a stateful forwarding table

P^{3}R is a protocol that combines the stateless procedure of next hop calculation and stateful forwarding strategy that relies on the forwarding table. As path connectivity can usually hold for a short time and continuous calculating results is likely to be the same, it is worth maintaining a stateful forwarding table for guaranteeing packet delivery and saving calculation cost. When forwarding a packet, the node first looks up its forwarding table. If hit an valid entry, the packet will be forwarded as the entry. Otherwise, the algorithm of next hop calculation will be triggered. Therefore, the forwarding table management and entry validness examining are significant. We can abstract the forwarding table to a cache whose key is the destination node ID, and the most important parts of it are the admission and eviction policies as follows.

1) Admission policy: the forwarding table entries have three sources, namely reverse path, greedy mode, and perimeter mode. To take these cases differently, we set a TYPE field in each forwarding table entry to indicate its origin and assign different priorities according to TYPE field. Entries from reverse paths (TYPE $= 0$) are built when receiving any packet. E.g., whenever node A receives a data packet from B (or a beacon message from S), with source S, it will create a forwarding entry whose destination is S and next hop is B (or S). The reverse path has the highest priority, because it is an existing and stable path to the source node. TYPE $= 1$ means the entry is from perimeter mode, and it has the lowest priority. This is because the perimeter mode is the last choice facing the dilemma where no neighbor is closer to the destination than the forwarding node. TYPE $= 2$ or $j (j \geq 3)$ are all from the greedy mode with the same priority between the reverse path and perimeter mode, but they mean the node is not at a crossroad or at crossroad j (crossroads are encoded from number 3) respectively. Now, we propose the admission policy in P^{3}R: whenever the key of the new entry is not hit in the table, add it into the table; when the hit entry has the same next hop and TYPE, reset the timer of it; when the hit entry has a different next hop or TYPE, replace the hit entry if its priority is lower than the new one.

2) Eviction policy: the forwarding table entry is evicted when its timer runs out or fails in the validness examining. The latter aims to check if the next hop is still a neighbor of the current node and can deliver the packet to the destination, which should have both accuracy and low complexity. Hence, our trajectory prediction and crossroad recognition algorithms are involved to increase the sensitivity to topology changes. The steps of next hop validness examining are as follows. First, we check whether the forwarding table entry and corresponding neighbor entry expire. If so, the entry is thought to be invalid. Second, P^{3}R takes different tests according to the entry TYPE: if the entry is built from a reverse path, it can remain to be valid until the expiration time; if it is from perimeter mode, the test of right hand principle shall be executed; if it is at crossroad j, check whether it is still in the coverage of crossroad j currently; otherwise, it is built

from greedy mode but not at a crossroad. In this case, the entry is still worth relying on if the current location of the node is towards the destination, i.e., the direction angle keeps unchanged or decreases.

IV. Simulation

We simulate P^{3}R on the NS-3 simulator. The coefficient $\alpha_1, \alpha_2, \alpha_3$ are set to be equal as default. The timers of forwarding table entry and neighbor entry are set to be equal to the beacon cycle. We compare P^{3}R with three typical map-independent protocols: two classical protocols, GPSR and AODV; and one of the state-of-the-art protocols, GPCR. AODV simulations run on NS-3 built-in module. We realize GPSR and GPCR under the framework of P^{3}R.

At the physical layer, all simulations use 802.11a[1] radios operating at a fixed rate of 6 Mbps with OFDM. In addition, to avoid the problem of hidden stations, the full techniques of 802.11a such as RTS/CTS mechanism are enabled. We use UDP protocol in the transport layer in all simulations. In the real network, the bidirectional flows such as Client/Server model are dominant, so we only use bidirectional flows in the application layer. We set ten nodes as requesters, ten nodes as responders and others as forwarding nodes. Each requester sends a request packet with a 64-byte payload to a responder every fixed time, and the responder will send a reply packet with 1000-byte payload back when receiving a request packet.

We obtained vehicle traces on real roads of Zhongguancun, Beijing, China using SUMO, where 30 to 100 vehicles were moving at a velocity varying from 8 m/s to 20 m/s in different experiment sets. The size of the scenario is about 1500 m by 900 m, and the transmit power is 1 W, which is strong enough to ensure that there is no isolated subnet. In order to reduce the impact of random fluctuations, we ran each simulation ten times with different random seeds and took the mean of each simulation experiment. Each value presented in subsequent figures is the mean of 10 simulation runs. For comparisons, we introduce four quality metrics as follows:

Round trip time: the amount of time from a requester sending a request to it receiving the corresponding reply, which describes the delay of bidirectional communication.

Round trip delivery ratio: the total number of reply packets received by a requester divided by the total number of request packets it sent, which describes the packet delivery ratio of bidirectional communication.

Overhead: The ratio of the total number of bytes in the packet header, protocol packets, and unsuccessfully delivered packets to the total number of bytes of application layer data in the successfully delivered packet. For example, if overhead is 0.6, it means you will waste 0.6 bytes for successfully delivering one byte of application layer data. Let $S(m,i)$ denote the i-th packet sent to MAC layer of node m and $R(m,i)$ describe the i-th packet received in application layer of node m. The overhead can be defined as follows:

$$Overhead = \frac{\sum_{m=1}^{M}\sum_{i=1}^{S_m} B_{S(m,i)}}{\sum_{m=1}^{M}\sum_{i=1}^{R_m} PB_{R(m,i)} \cdot H_{R(m,i)}} - 1 \qquad (4)$$

where M denotes the number of nodes; S_m, the total number of packets sent to MAC layer of node m; R_m, the total number of packets received in application layer of node m; B, the packet bytes; PB, the payload bytes; H, the packet hops.

A. Network size

Fig. 4 shows the performance of AODV, GPSR, GPCR and P^{3}R on different network sizes. The performance of the round trip delivery ratio of the four protocols is shown in Fig. 4(a). We can see that the ratio of P^{3}R is relatively stable in the range of 75% to 80% and it surpasses AODV when the number of nodes is more than 50. This is because the effective next hop selection of P^{3}R ensures the connection between end to end as the neighbor number increasing. However, for AODV, more nodes need more time to get route convergence, i.e., more packets will wait in the local queue in the path repair state. Since the queue length is limited, more packets will overflow from the queue and packet loss ratio increases. GPCR and GPSR have the lowest two delivery ratios, due to their unintelligent mechanism of next hop selection. In a word, P^{3}R achieves high round trip delivery ratio with node number increasing.

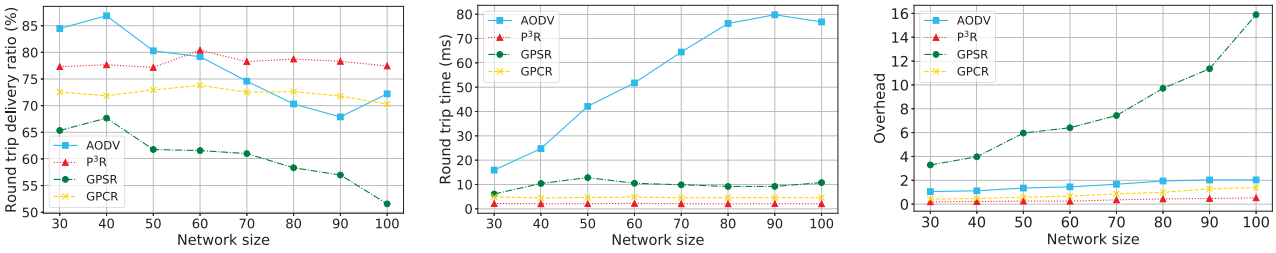
Fig. 4(b) shows the trends of the round trip time. Obviously, the round trip time of AODV is much longer than that of GPSR, GPCR, and P^{3}R, and keeps linear growth. This is because the packets waiting in the local queue cannot be sent until the end of the repair state in AODV. GPSR, GPCR, and P^{3}R calculate next hop directly and send the packet to the selected node without waiting in the queue, but the time of P^{3}R is only about 2 ms, and 1/2 of GPCR. This is because GPCR always selects nodes on the crossroads conservatively, which usually leads to more hops and longer delivery time on real roads.

Fig. 4(c) shows the overhead of the four protocols with different network sizes. All the four lines increase with node number increasing, but their growth rates are totally different. GPSR has the largest overhead and growth rate, mainly because of its low packet delivery ratio. The reason for AODV being the second is twofold. First, AODV will broadcast more protocol packets in path repair state with node number increasing. Second, declining packet delivery ratio of AODV exacerbates this trend. Finally, depending on high and stable packet delivery ratio as well as controlled beacon messages that are proportional to the number of nodes, P^{3}R produces less overhead than the other protocols.

B. Velocity

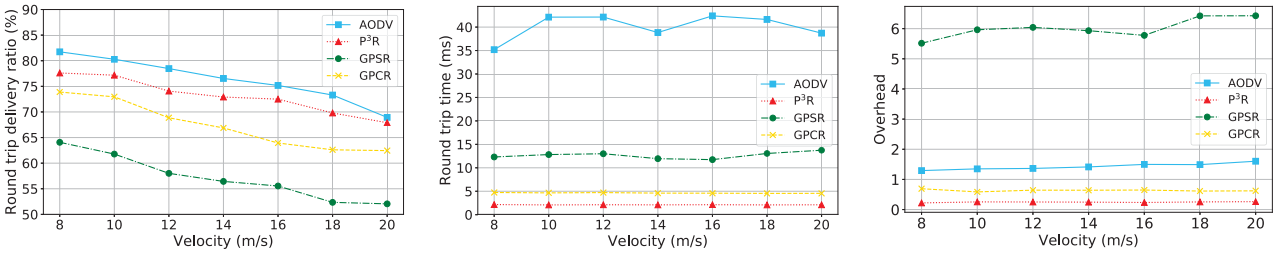
The performance of AODV, GPSR, GPCR, and P^{3}R with different mean velocities is shown in Fig. 5. The network size is set to 50, and mean velocity varies from 8m/s to 20m/s.

[1]To facilitate simulations, we use 802.11a instead of 802.11p, which will not influence the correctness and performance of protocols



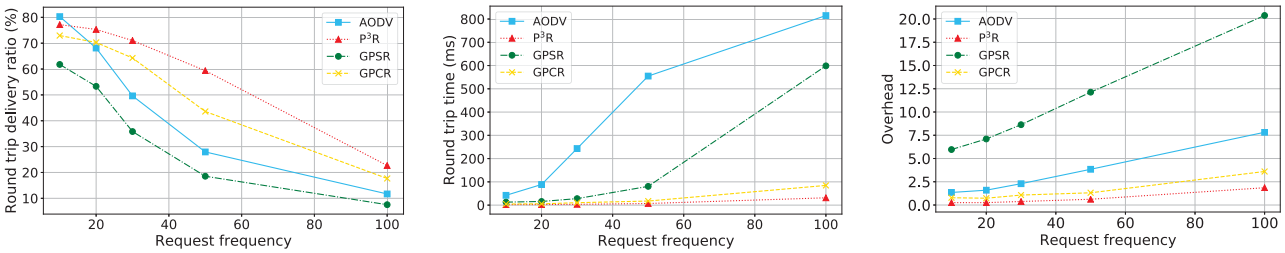
(a) Round trip delivery ratio (b) Round trip time (c) Overhead

Fig. 4: Performance on network sizes. Velocity is 10m/s and request frequency is 10/s



(a) Round trip delivery ratio (b) Round trip time (c) Overhead

Fig. 5: Performance on velocity. Network size is 50 nodes and request frequency is 10/s



(a) Round trip delivery ratio (b) Round trip time (c) Overhead

Fig. 6: Performance on request frequency. Network size is 50 and velocity is 10m/s

Fig. 5(a) shows the round trip delivery ratio of these four protocols. Obviously, all lines represent a downward trend, and the performance of P^{3}R is between AODV and GPCR, which is consistent with the result in Fig. 4(a) when the number of nodes is 50. While, it is noted that the distance between P^{3}R and AODV shrinks with velocity, which indicates P^{3}R is more stable than AODV with velocity increasing. For round trip time shown in Fig. 5(b), P^{3}R is still the lowest and most stable protocol, while AODV and GPSR show a more fluctuating characteristic. As shown in Fig. 5(c) both GPSR and AODV see an increasing trend on overhead and are much higher than P^{3}R. This is because the change in velocity brings great randomness to the topology, Although GPCR is also stable with speed increasing, it is higher than P^{3}R due to lower delivery ratio.

C. Request frequency

We conduct different request frequency to figure out the scalability of these four protocols with the application tasks growing. As Fig. 6 shows, P^{3}R is much more scalable and stable than the others in delivery ratio, round trip time, and overhead. Although P^{3}R is a little lower than AODV at the beginning in Fig. 6(a), it keeps higher than others as the channel becomes increasingly crowded. In contrast to the 800 ms delay of AODV in Fig. 6(b) and 20 overhead of GPSR in Fig. 6(c) when frequency is 100, which is unacceptable for delay-sensitive applications, P^{3}R is much better and more stable, benefiting from the lower overhead that only contains small beacon protocol packets, and efficient next hop calculation that ensures the connectivity with a little cost.

V. Implementation

We implement P^{3}R on commercial workstations (Dell M4800) and the MK5 On-Board Units (OBU) devices [19]. MK5 OBU is an advanced VANET OBU designed by Cohda Wireless and equips with the ubuntu 14.04 operating system. It supports 802.11p/WAVE(Wireless Access in the Vehicular Environment) protocol and provides meter-level positioning. The

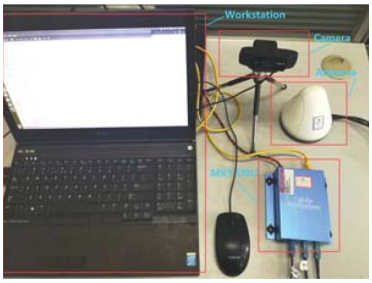


Fig. 7: Device setup in a car node

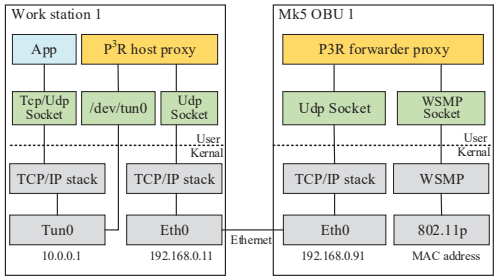


Fig. 8: The system framework

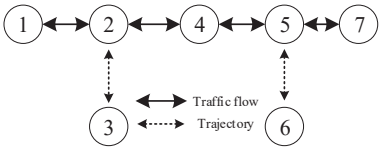


Fig. 9: The topology of the seven car nodes in the implementation

SDK of MK5 OBU provides WSMP (WAVE Short Message Protocol) socket, which is similar to the general sockets for developers, such as *sendto* and *receivefrom* functions. Fig. 7 shows the devices in each vehicle node, containing an MK5 OBU with an antenna supporting 802.11p physical standards, a workstation responsible for running various applications, and a high definition camera. The MK5 OBU is connected with the workstation by an Ethernet cable, and the MK5 OBUs are connected by 802.11p wireless.

We realize P^{3}R in the user space of workstation and MK5. At the ingress edge node, we run proxy process to encapsulate the original packet with P^{3}R protocol information. Then the encapsulated packet will be forwarded hop-by-hop by P^{3}R until reaching the egress edge node. In detail, the P^{3}R protocol structure is shown in Fig. 8, which consists of two parts, systems on the workstation and MK5 OBU. Both P^{3}R proxies on workstation and MK5 run in the user space and use UDP socket to communicate with each other. Differently, the proxy on workstation uses TUN (TUNnel) interface to encapsulate/decapsulate packets, while the proxy on MK5 uses WSM socket to forward packets among vehicles.

Our testbed consists of seven car nodes shown in Fig. 9, and each of them is equipped with such a suite of devices shown in Fig 7. We run EasyRtc [20], a real-time multi-party video conference application, on the stationary nodes 1, 4, and 7, and each MK5 OBU runs P^{3}R as mentioned above. The results show they experience near non-interruption and high-quality video conference when node 2,3,5, and 6 keep moving up and down within an appropriate range. The results show that P^{3}R performs much better than GPSR, achieving 98.6% delivery ratio and 7.33 ms handoff time in this scenery. Therefore, P^{3}R indeed improves the communication connectivity in VANET.

VI. Conclusion

For VANET, this paper proposes P^{3}R protocol, an efficient routing protocol based on trajectory prediction and crossroad recognition. Simulation and implementation results show P^{3}R not only has high packet delivery ratio, lowest delay and overhead compared with AODV, GPSR, and GPCR, but also is more stable and scalable with the topology and workload changing.

Acknowledgment

This paper is supported by NSFC (68172213, 61432009) and Huawei Innovation Research Program (HIRP). Bin Liu is the corresponding author.

References

[1] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," Tech. Rep., 2003.
[2] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Tech. Rep., 2003.
[3] B. Karp and H.-T. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks," in *Proceedings of ACM MOBICOM*, 2000.
[4] J. Zhao and G. Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," *IEEE transactions on vehicular technology*, vol. 57, no. 3, pp. 1910–1922, 2008.
[5] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea, "Vanet routing on city roads using real-time vehicular traffic information," *IEEE Transactions on Vehicular technology*, vol. 58, no. 7, pp. 3609–3626, 2009.
[6] Y. Li, Z. Wang, C. Zhang, H. Dai, W. Xu, X. Ji, Y. Wan, and B. Liu, "Trajectory prediction algorithm in vanet routing," *Journal of Computer Research and Development*, vol. 54, no. 11, pp. 2421–2433, 2017.
[7] G. Xue, Y. Luo, J. Yu, and M. Li, "A novel vehicular location prediction based on mobility patterns for routing in urban vanet," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 222, 2012.
[8] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *Proceedings of ACM SIGCOMM*, 1994.
[9] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile computing*, 1996, pp. 153–181.
[10] Z. J. Haas and M. R. Pearlman, "The performance of query control schemes for the zone routing protocol," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, pp. 427–438, 2001.
[11] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, "Geographic routing in city scenarios," *ACM SIGMOBILE mobile computing and communications review*, vol. 9, no. 1, pp. 69–72, 2005.
[12] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A distance routing effect algorithm for mobility (dream)," in *Proceedings of ACM MOBICOM*, 1998.
[13] V. Naumov and T. R. Gross, "Connectivity-aware routing (car) in vehicular ad-hoc networks," in *IEEE INFOCOM*, 2007, pp. 1919–1927.
[14] I. F. Akyildiz and W. Wang, "The predictive user mobility profile framework for wireless multimedia networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 6, pp. 1021–1035, 2004.
[15] V. Namboodiri and L. Gao, "Prediction-based routing for vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 4, pp. 2332–2345, 2007.
[16] S. Jain, K. R. Fall, and R. K. Patra, "Routing in a delay tolerant network," *acm special interest group on data communication*, vol. 34, no. 4, pp. 145–158, 2004.
[17] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi, "Mobeyes: smart mobs for urban monitoring with a vehicular sensor network," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 52–57, 2006.
[18] R. Aissaoui, A. Dhraief, A. Belghith, H. Menouar, H. Mathkour, F. Filali, and A. Abudayya, "Hcbls: A hierarchical cluster-based location service in urban environment," *Mobile Information Systems*, vol. 2015, pp. 1–16, 2015.
[19] Cohda wireless, "mk5 obu". [Online]. Available: http://www.cohdawireless.com/solutions/hardware/mk5-obu
[20] Easyrtc, "the fastest way to build your own webrtc". [Online]. Available: https://easyrtc.com