Offloading Elastic Transfers to Opportunistic Vehicular Networks Based on Imperfect Trajectory Prediction

***Abstract*—Due to the high cost of cellular networks, vehicle users would like to offload elastic traffic through vehicular networks as much as possible. This demand prompts researchers to consider how to make the vehicular network system achieve better performance for requests coming online, such as maximizing throughput. The traffic in vehicular networks is transferred through opportunistic contacts between vehicles and infrastructures. When making scheduling decisions, the scheduler must be aware of vehicles' future trajectories. Vehicles' future trajectories are usually predicted by trajectory prediction algorithms when users are not willing to report their future trips. Unfortunately, no trajectory prediction algorithm can be completely accurate, and these inaccurate prediction results will reduce the throughput. In this paper, we focus on overcoming this challenge. Specifically, we first measure the LSTM (Long Short Term Memory) prediction method that has been widely used recently and understand the accuracy of predicted contacts. Then, based on the enlightenment from the measurement, we design a system, *i.e., i-Offload*, to offload elastic traffic under an imperfect trajectory prediction algorithm. The experimental results show that our method still has good throughput and scheduling efficiency under an imperfect prediction algorithm. Compared with an existing scheduling method, our method outperforms it by 42% and 45% in terms of scheduling throughput and scheduling efficiency, respectively.**

***Index Terms*—vehicular network, file downloading, imperfect trajectory prediction, LSTM**

I. Introduction

With the rapid development of wireless communication technology and increasing vehicle ownership, it is becoming common for people to access the Internet through vehicles. According to Cisco's report [1], Machine-to-Machine connections will be half of the global connections by 2023, in which connected vehicles will be the fastest-growing category. The rapidly increasing number of vehicle applications, such as fleet management, in-vehicle entertainment systems, and navigation, will lead to a tremendous increase in Internet traffic.

In vehicular networks, vehicles can obtain application data from the Internet in two ways. The first way is from cellular networks. This kind of transmission has a wider coverage because of its widespread infrastructures. However, the bandwidth is limited and expensive. The second way is from Road Side Units (RSUs) or other vehicles. RSUs are proposed to be deployed along roads, and they are connected to the Internet directly via wired links. When one vehicle requests content from the Internet, one or multiple RSUs can prefetch the content. Vehicles equipped with on-board units can get the content directly from the RSU via Vehicle-to-Infrastructure (V2I) communication or obtain the content from other vehicles via Vehicle-to-Vehicle (V2V) communication. Dedicated Short Range Communication (DSRC) [2] or LTE [3] technology enables these two types of communications. The coverage radius of an RSU is usually a few hundred meters. Due to the sparse deployment of RSUs, the coverage of RSUs is not continuous. This kind of method has a low cost, but the connections between vehicles and RSUs are opportunistic.

As the bandwidth of cellular networks is limited and expensive, people would like to offload traffic to vehicular networks as much as possible. The traffic requested by vehicles can be divided into two categories roughly according to their time sensitivity. The first type is *interactive* data, which requires a very small transmission delay, such as road safety signals. The other type is *elastic* data, which is less stringent in terms of time delay, such as multi-media files and road maps. In this work, we focus on elastic requests since interactive requests should be served immediately to satisfy their critical requirement in response time. As a result, the transmission path for interactive requests should be determined directly, and it is unnecessary to consider these requests in the scheduling problem.

However, the high mobility of vehicles makes it hard to offload elastic traffic efficiently through vehicle networks. The mobility of vehicles and the limited coverage of a single RSU make network connections opportunistic. Therefore, to make data transfer decisions through such opportunistic connections, it is necessary to know when these opportunistic connections will occur, *i.e.*, the exact trajectories of vehicles in the future. There are two approaches to realize this step in existing works. The first one assumes that the motion pattern of vehicles obeys some distribution or assumes that vehicles will report their future trajectories [4]. The other one obtains future trajectories by using trajectory prediction algorithms and assumes that the prediction results are accurate. For example, systems in [5] and [6] use the prediction results generated by trajectory prediction algorithms to select the most suitable relay vehicles.

However, both approaches face the same problem: the system cannot get a completely accurate knowledge of future trajectories in practice. Firstly, vehicles may not be willing to report their future trajectories due to privacy concerns. Even if they allow future trajectories reported, their trajectories may change at any time, making their previous reports invalid. Secondly, prediction results are not always completely accurate, and wrong prediction results will result in the failure of scheduling decisions.

Therefore, it is necessary to design a system that can efficiently carry out traffic offloading under imperfect trajectory prediction algorithms. As far as we know, there are few works in this field at present, and the most representative one is [7],

where the author proposes a model to express and account for the inaccuracy of prediction results. They use the Markov prediction method to verify their model, but this model's validity under new trajectory prediction methods, such as the LSTM method, is still missing.

In this paper, we propose i-Offload to offload elastic traffic under imperfect trajectory prediction algorithms. Our goal is to maximize the throughput of vehicular networks, and our main contributions are as follows:

- We first measure the prediction results of the LSTM prediction method, which has been widely used recently. It helps us understand where incorrect predictions are likely to occur and deal with these incorrect predictions properly. To the best of our knowledge, we are the first to study the accuracy of predicted contacts for the LSTM method.
- Then, based on the enlightenment from the measurement experiments, we design the i-Offload system to offload elastic traffic under imperfect trajectory prediction algorithms. In this system, we propose an adaptive scheduling window determination method to reduce the impact of imperfect predictions and a strategy to make scheduling decisions on multiple transmission paths.
- Finally, we point out some shortcomings of an existing method for simulating prediction results with different accuracy. The modified model is called dens-fog, and it can simulate a series of prediction results with a specified accuracy, which can be used to evaluate scheduling algorithms' performance.

Our simulation experiments show that, compared to the MMCD method [8], under an imperfect trajectory prediction environment, our method outperforms it by 42% and 45% in terms of scheduling throughput and scheduling efficiency, respectively. Besides, our system can also achieve better throughput and scheduling efficiency in early-deployment scenarios of vehicular networks.

The rest of the paper is organized as follows. Section II reviews related works. Section III describes our problem. We measure the accuracy of trajectory prediction results of the LSTM prediction method in Section IV. We describe the i-Offload system in Section V. The dense-fog model, which is used to generate a series of prediction results for simulations, is proposed in Section VI. We evaluate the scheduling performance of the i-Offload system in Section VII. Finally, we summarize our work in Section VIII.

II. Related Work

A. Trajectory Prediction in Vehicular Networks

Trajectory prediction algorithms predict the future coordinates of vehicles in our scenario. Therefore, we need to have a comprehensive understanding of trajectory prediction algorithms used in vehicular networks.

Xie *et al.* [9] classify existing trajectory prediction methods into two categories: model-based prediction and data-driven prediction. Model-based prediction methods are based on Kinematics analysis or statistical models [10]–[12]. These methods are more explanatory, but they are only applicable to predict for a short period. In contrast, data-driven prediction approaches [13]–[15] are getting more and more attention from researchers, which use a learning method to generate the future coordinate sequence. The historical trajectory data of vehicles are used for model training, and then the trained model is used for trajectory prediction. This kind of method is considered superior to the model-based prediction method in prediction accuracy [9].

Most of them aim at designing a high accuracy prediction algorithm. However, research on the impact of prediction accuracy on communication between vehicles is still lacking, especially for data-driven prediction methods. Therefore, in this paper, we measure the LSTM-based trajectory prediction algorithm to understand this data-driven method's prediction accuracy.

B. Traffic Offloading Methods in Vehicular Networks

Offloading elastic traffic in vehicular networks is a promising but challenging task [16]. Due to each contact's short duration, the transmission for a large file cannot be completed through a single connection. The solution to this problem is to divide the large file into small segments. In this process, network coding technology is proved to be an effective way to reduce scheduling complexity and improve scheduling efficiency [17], [18]. With network coding, each content segment will be divided into several blocks, *i.e., original blocks*, and then the scheduling system will use these original blocks to generate coded blocks. The target vehicle can recover the initial segment as long as it receives a sufficient number of independent coded blocks. It does not matter whether a particular block is delivered successfully. This feature can reduce the scheduling complexity and improve scheduling efficiency. Therefore, we use this method to process content into small segments and generate coded blocks before scheduling.

The scheduling algorithms can be divided into two categories according to whether roadside infrastructures are used when offloading traffic. The first category is called *infrastructure-free* offloading [19]–[22], where the vehicular network scenario does not include roadside infrastructures. In this work, we study the problem of traffic offloading with the help of roadside infrastructures. Therefore, this category is different from our scene and does not apply to our problem.

We refer to the works that the infrastructures are involved in traffic offloading as *infrastructure-assisted* offloading. According to whether the contents requested by different targets are the same or not, these studies can be further divided into two categories. One of them is *redundant* content offloading [23]–[25], in which most of the vehicles in the system have the same target file. The other is *heterogeneous* content offloading, in which vehicles request different files, and our work falls into this category.

A distinction in this kind of works is how to know future trajectories of vehicles. As the contact is opportunistic, the scheduler must know vehicles' future trajectories to make scheduling decisions. Yoon *et al.* [4] propose a system to maximize the amount of requested data that can be delivered to the target vehicles during their journeys. They assume that

the time at which vehicles arrive at each RSU is known. In this way, the scheduler can find all transmission paths. Yang *et al.* [26] combine V2I and V2V communications to carry out the scheduling. They assume that trajectories can be predicted accurately. They propose a scheduling strategy whose insight is to select the path with the highest rate from all usable paths for transmission. However, this approach only compares the priority between usable paths, ignoring the competition between different requests. This may cause some requests to starve to death, *i.e.*, requests with low rate usable paths may never be satisfied. This has been improved by Ota *et al.* [8]. They design a maximum rate and minimum delay method (MMCD). This method first sorts the requests according to their deadline and then prioritizes the transmission path with a large capacity when choosing the transmission path.

The above papers assume that vehicles' trajectories are known or can be predicted accurately in advance. However, prediction results may be incorrect, that is, there is no way for the scheduler to know real trajectories in advance. This is also the most significant difference between our work and the above papers.

Wang *et al.* notice this problem in [27]. They focus on the tradeoff between the downloading delays and the cost for file downloading in VANETs. In their work, they propose an adaptive routing method to solve this problem, where a vehicle might re-estimate encountering time and do a re-decision if the vehicle does not encounter RSUs as expected. Although this work has noted the problem of imperfect prediction, their goal is to maximize the cost-efficient of offloading, which is different from our goal. In our paper, the goal is to maximize the throughput, which is the same as [7]. Malandrino *et al.* propose a model in [7], *i.e. fog-of-war*, which is used to simulate prediction results with different accuracy. Given a contact, this model can output a probability value to represent the possibility that this contact will occur in the future. They use these probability values as edge weights of a time expansion graph, and the amount of data to be prefetched by each RSU is determined by solving an optimization problem formulated on this graph.

In their paper, the inaccuracy of prediction is also considered, but some of the assumptions used to construct the fog-of-war model may not hold in other trajectory prediction algorithms. We will show this in Section VI. Their scheduling method relies on the fog-of-war model's output results, so it does not apply to our problem. By contrast, we first measure the LSTM trajectory prediction algorithm's prediction results to show the pattern of the inaccuracy of results. Based on the inspiration from the measurement results, we then design an offloading system, *i.e.*, i-Offload, to solve the problem.

III. Problem Description and Analysis

A. Problem Description

Consider the scenario shown in Figure 1. Assume there have been some Roadside Units (RSUs) deployed by the roadside, denoted by $\mathbb{U}_j$ $(j = 1, 2, \dots)$. These RSUs are with the function of storage and they are connected to the Internet directly via wired links. Many vehicles, denoted by $\mathbb{V}_k$ $(k = 1, 2, \dots)$, are traveling around the city, and they are equipped with communication modules that enable short-range communication between RSUs and vehicles. These vehicles may have numerous data transfer requests to download various files from the Internet. The vehicle that initiates the request i is denoted by $\mathbb{R}_i$, and we have $\mathbb{R}_i \in \{\mathbb{V}_k\}$.

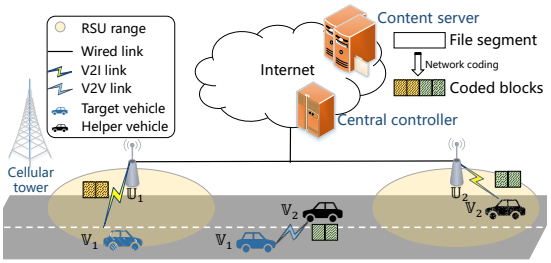


Fig. 1: Data transmission in a vehicle network.

A vehicle can communicate with one RSU/vehicle when it moves into the transmission range of the RSU/vehicle. Conventionally, the event that two entities move into the transmission range of each other is referred to as a *communication contact*. A vehicle-RSU (*i.e.*, V2I) link or a vehicle-vehicle (*i.e.*, V2V) link can be used for data transmission only when the relevant communication contact occurs. The duration of each communication contact is limited, which means the amount of data that can be transmitted within this contact is limited. Therefore, a requested file should be divided into blocks, and then blocks can be transmitted over different paths formed by a series of communication contacts.

Let us illustrate the data transmission in a vehicle network using the following example. Assume $\mathbb{V}_1$ initiates a request at t_0. The requested file is split into 4 blocks. $\mathbb{U}_1$ prefetches two blocks at t_1, and $\mathbb{U}_2$ prefetches the other two blocks at t_2. Later, the target vehicle $\mathbb{V}_1$ passes by $\mathbb{U}_1$ at t_3 $(t_3 > t_1)$ and get two blocks from $\mathbb{U}_1$. Moreover, one vehicle $\mathbb{V}_2$ passes by the RSU $\mathbb{U}_2$ at t_4 $(t_4 > t_2)$ and it retrieves the two blocks from $\mathbb{U}_2$. $\mathbb{V}_2$ later relays these two blocks to the target vehicle $\mathbb{V}_1$ when two vehicles meet with each other at t_5 $(t_5 > t_3)$.

In this example, we see two paths involved. The first path is $(\mathbb{U}_1, \mathbb{V}_1)$ wherein the communication contact occurs at t_3; and the second path is $(\mathbb{U}_1, \mathbb{V}_2, \mathbb{V}_1)$ wherein the communication contacts occur at t_4 and t_5 respectively. There is no V2V link on the first path, for which we name it as a *V2I path*. The second path is named as a *hybrid path* because both V2I and V2V links are involved. We can see that there is one and exactly one RSU on each path, which is responsible for getting blocks from the content server in the Internet.

Mathematically, let $\mathcal{X}_{i,n}$ $(n = 0, 1, \dots)$ denote the n-th path for the request i. The l-th node on this path is denoted by $\mathcal{X}_{i,n}^l$. Please note that $\mathcal{X}_{i,n}^l$ is a RSU when $l = 0$ and it is a vehicle (either a relay vehicle or the target vehicle) when $l >= 1$. The time period during which the l-th node sends out the relative blocks to the next hop is denoted by $\mathcal{T}_{i,n}^l$, and there must be $\mathcal{T}_{i,n}^l < \mathcal{T}_{i,n}^{l+1}$. Furthermore, we denote the number of hops on the path $\mathcal{X}_{i,n}$ as $L_{i,n}$.

As we have explained, two entities can exchange data packets only they move into the transmission range of each

other. Let $p_t^{u,v}$ indicate whether there is a communication contact between two entities (u, v) all the time during the period t. For now, $p_t^{u,v}$ can be 0 or 1. Mathematically, whether the path $X_{i,l}$ really exists and can be used for data transmission can be calculated as follows,

$$\mathcal{P}_{\mathcal{X}_{i,n}} = \prod_{0 \le l < L_{i,n}} p_{\mathcal{T}_{i,n}^l}^{\mathcal{X}_{i,n}^l, \mathcal{X}_{i,n}^{l+1}}. \tag{1}$$

The resource of each vehicle or each RSU is limited, therefore the requests in the network have to compete for resources and it is not guaranteed that all requests can be satisfied by the vehicle network. The base stations of cellular networks, which are supposed to provide seamless Internet service, serve as the last resort to complete the remaining blocks that cannot be transmitted by the vehicle network.

Generally said, people always want to offload their traffic demands to vehicle networks (including RSUs and vehicle ad hoc networks) as much as possible because cellular networks are expensive and bandwidth-limited. The problem here is how to find the optimal scheduling solution for all requests, *i.e.*, feasible transmission paths for blocks of requests that can maximize the amount of offloaded traffic. Mathematically, this scheduling problem can be formulated as Equation 2.

$$\max_{\mathcal{X}_{i,n}, \mathcal{Q}_{\mathcal{X}_{i,n}}} \sum_i \sum_n \mathcal{P}_{\mathcal{X}_{i,n}} \mathcal{Q}_{\mathcal{X}_{i,n}} \tag{2}$$

Here, $\mathcal{Q}_{\mathcal{X}_{i,n}}$ is the amount of data that is transmitted via the path $\mathcal{X}_{i,n}$. To ensure the feasibility of the solution, the following constraints must be satisfied.

- First, RSU $\mathbb{U}_j$ can serve vehicles at a maximum total transfer rate of $\mathcal{B}_{\mathbb{U}_j}$. We assume that RSUs have sufficient buffering memory and also have sufficient bandwidth with the Internet, therefore the total upstream bandwidth (transfer rate to vehicles) is the only resource bottleneck for a RSU. Mathematically, the constraint is as follows.

$$\sum_{\substack{\mathcal{X}_{i,n}^0 = \mathbb{U}_j \\ \& \, \mathcal{T}_{i,n}^0 = t}} \mathcal{Q}_{\mathcal{X}_{i,n}} \le \mathcal{B}_{\mathbb{U}_j}, \qquad \forall t, j \tag{3}$$

- Second, a vehicle can only communicate with one vehicle at the same time. We assume that all V2I and V2V communications are carried out through different antennas so that the traffic between V2V and V2I does not interfere with each other. However, for V2V communication, in general, we assume that vehicles cannot transmit and receive at the same time. Also, a vehicle should not receive transmissions from multiple vehicles at the same time to avoid transmission conflicts. Mathematically, this one-to-one transmission constraint is represented as follows.

$$\left|\{\mathcal{X}_{i,n} \mid \exists l, X_{i,n}^l = \mathbb{V}_k \,\&\, \mathcal{T}_{i,n}^l = t\}\right| \le 1, \quad \forall t, k \tag{4}$$

$$\left|\{\mathcal{X}_{i,n} \mid \exists l \ge 2, X_{i,n}^l = \mathbb{V}_k \,\&\, \mathcal{T}_{i,n}^{l-1} = t\}\right| \le 1, \forall t, k \tag{5}$$

- Third, the amount of data that is transmitted successfully for request i should not exceed its demand. Mathematically, let Q_i is the size of the demand of the request i, then we should have

$$\sum_n \mathcal{P}_{\mathcal{X}_{i,n}} \mathcal{Q}_{\mathcal{X}_{i,n}} \le Q_i, \qquad \forall i \tag{6}$$

If all requests of these vehicles and all communication contacts among vehicles and RSUs can be known exactly in advance, a centralized algorithm can solve the above scheduling problem and derive the optimal schedule.

However, vehicles may not be willing to report their future trajectories due to privacy concerns. Even they allow future trajectories reported, their trajectories may change at any time, which makes their previous reports invalid. Therefore, communication contacts cannot be known in advance accurately. Similarly, vehicles can submit requests at any time, and the arrivals of these requests cannot be predicted accurately either. Besides these two factors, the computational complexity for solving the above optimization problem must be considered, as the scheduling decision should be determined as soon as possible to make the system work well in an online manner.

In this paper, we will study the scenario described above to propose an online scheduling algorithm for unpredictable requests arriving from vehicles with unknown trajectories.

B. Problem Analysis

To improve the optimality of the scheduling decision, we should try to learn as much as possible about requests and vehicles and exploit the information we can learn as much as possible.

1) being deadline-aware: In this work, we focus on elastic requests since interactive requests should be served immediately to satisfy their critical requirement in response time. As a result, the transmission path for interative requests can be determined directly and it is unnecessary to consider these requests in the scheduling problem. In contrast, there can be a large number of candidate paths to transmit data for elastic requests because their requirement on response time is less strict. The scheduler needs to choose the optimal paths for blocks of each request.

It can be required that all elastic requests should be submitted together with their individual deadline information. The deadline information of elastic requests is significantly valuable for us to improve the scheduling efficiency. Intuitively, when two requests are competing for resources, the far-deadline request can be postponed to make room for the request with an urgent deadline. We assume that we know the detailed information of each request i, *i.e.*, a four-tuple $(\mathbb{R}_i, ts_i, td_i, Q_i)$, wherein $\mathbb{R}_i$ is the vehicle that initiates the request i and we name it as the *target* of the request; ts_i and td_i are the start time (from which the requested file can be transmitted) and the deadline of the request (after which the transmission is useless for the request); and Q_i is the demand size of the request.

2) using imperfect trajectory prediction results: If we can predict vehicle trajectories accurately, we will be able to predict whether two entities can contact each other for data transfers at any particular time point t, and then we can arrange transmission paths efficiently.

Trajectory prediction is an active research area and quite a few prediction algorithms have been proposed. At a time point t, a prediction algorithm can predict the geographical location of any individual vehicles over a period of time $t + \Delta(\forall \Delta >$

0). Unfortunately, none of them can guarantee the accuracy of their prediction results [28].

Obviously, an incorrect prediction of a vehicle's future location may have a negative effect on the performance of the scheduling system, *i.e.*, lowering the amount of traffic that can be offloaded successfully.

We can illustrate its negative effect using the following simple example. Assuming that a vehicle $\mathbb{V}_1$ requests a file to be downloaded from the Internet, and the vehicle is predicted by an algorithm to be in the transmission range of a RSU $\mathbb{U}_1$ during a time period $\mathcal{T}_1$. The scheduler thus plans a path for this request, that is, $\mathbb{U}_1$ prefetches the file from the Internet before $\mathcal{T}_1$, stores the file and forwards it to $\mathbb{V}_1$ during $\mathcal{T}_1$. In order to realize the planned data transfer, $\mathbb{U}_1$ needs to consume its bandwidth from the Internet, and it also needs to reserve the bandwidth to vehicles during $\mathcal{T}_1$. However, the vehicle $\mathbb{V}_1$ does not appear at the predicted location during $\mathcal{T}_1$. After the scheduler finds that the planned path fails, it has to find available resources to plan other paths to satisfy the request of $\mathbb{V}_1$. It can fail if no resource available before the request's deadline, which may not happen if the scheduler can start to plan paths for the request earlier. On the other hand, the serving bandwidth of the RSU is wasted because the reserved resource is not used. Definitely, we can see that the vehicle network can satisfy more requests if the incorrect prediction can be removed.

Since no prediction algorithm can guarantee that its prediction results are 100% correct, our scheduling algorithm must consider the negative effect of incorrect prediction results. We should try to make sure that it is able to work well with incorrect predictions from imperfect prediction algorithms, which is the most significant challenge we have to solve.

IV. Understanding the Performance of Trajectory Prediction Algorithms

As we mentioned above, we can arrange transmission paths for received requests efficiently if vehicle trajectories can be predicted accurately. Incorrect predictions of vehicles' future locations would definitely degrade our scheduling efficiency.

In this section, we will conduct a performance measurement study on a LSTM-based trajectory prediction algorithm, which is the most widely used method currently. Particularly, we will try to understand the accuracy of predicted contacts and the factors that have influences on the prediction accuracy. It helps us understand where incorrect predictions are likely to occur and deal with these incorrect predictions properly.

A. Experiment Settings

1) Trajectory prediction algorithm: As explained in II-A, data-driven trajectory prediction methods are regarded as more promising than model-based prediction methods. Among data-driven trajectory prediction methods, LSTM is considered to be particularly effective for time series prediction problems because it can retain the information from previous timeslots [29], and many trajectory prediction algorithms based on LSTM have been proposed in recent years [13], [15], [30]–[32]. Although these works make some modifications to the original LSTM network, these modifications are proposed to address their special concerns, and these works still share the same performance characteristic resulting from the original LSTM network. Therefore, in the following measurements, we focus on the original LSTM trajectory prediction method proposed in [13].

The authors of these works have conducted experiments to evaluate the performance of their own algorithms. But their concerns are whether the predicted location at any future time point is accurate, which is related to but different from our concerns. In this work, our concerns on the performance of prediction algorithms are the following two issues. First, whether a predicted contact (a contact that is predicted to occur by the prediction algorithm) really occurs, which is evaluated by *precision*. Second, whether all contacts that really occur can be predicted in advance by the prediction algorithm, which is evaluated by *recall*. We formally define two metrics as follows. Generally, if the distance of the two vehicles is close enough (less than 500 meters), we believe that communication can be conducted between these two vehicles, that is, there is a contact. However, the accuracy of coordinates is different from that of contacts. Think about this example, there are two vehicles and the predicted coordinates of both two vehicles deviate 1000 meters in the same direction relative to their real future coordinates. The distance between these two vehicles calculated from the predicted coordinates is the same as the real value. We can draw the correct conclusion that there is contact between these two vehicles, even if the prediction on the coordinates is not accurate. Therefore, it is necessary to analyze the accuracy of the contact not just the coordinates.

2) Metrics: Let N_g be the set of communication contacts that really occur, that is, the contacts in the ground truth dataset. The set of communication contacts obtained by the trajectory prediction algorithm is denoted as N_p. Some contacts in N_p really occur later as predicted, *i.e.* the predictions are correct, and we use N_{tp} to denote these correct prediction contacts.

The ***prediction precision rate*** is the ratio of the number of correctly predicted contacts to the number of all predicted contacts, *i.e.*, $p = |N_{tp}|/|N_p|$.

The ***prediction recall rate*** is the ratio of the number of correctly predicted contacts to the number of contacts that will really occur, *i.e.*, $r = |N_{tp}|/|N_g|$.

3) Model training: We use the *Scrg* dataset [33] to train and test a LSTM model. The model uses the locations of a vehicle in the past L timeslots as input to predict its location in future H timeslots, *i.e.*, L is the length of input sequences and H is the length of the output sequences in the LSTM model. The dataset consists of trajectories of 4,000 taxis within one day in Shanghai, and vehicle locations are collected at an interval of one minute.

We preproccess the dataset and select the trajectories of 1,600 taxis within 13 hours (08:00-20:59). The 13-hour period is divided into discrete timeslots of τ minute (thus $780/\tau$ timeslots in total), and the timeslot starting at 08:00 is defined as the first timeslot t_0. Therefore, the trajectory of a vehicle V_k can be extracted as a sequence $\mathcal{S}_k = \{(x_k^i, y_k^i)|i \in [0, 780/\tau - 1]\}$, where (x_k^i, y_k^i) denotes the longitude and

latitude of the vehicle V_k at the timeslot t_i. We take the trajectories within the first 11 hours (*i.e.*, $i \in [0, 660/\tau - 1]$) as the training dataset and get a trained model.

In the next subsection, we use the trained model to predict contacts within the remaining 2 hours, and compare the prediction results with the ground-truth (the contacts derived from the dataset) to evaluate the precision rate and recall rate.

B. Analyzing Prediction Results

We make predictions for each vehicle in each timeslot t_i ($i \in [660/\tau, 780/\tau - 1]$) within the testing dataset, *i.e.*, the locations $\{(x_k^i, y_k^i) | i \in [t - L + 1, t]\}$ are used by the LSTM model to generate a sequence $\{(x_k^i, y_k^i) | i \in [t + 1, t + H]\}$ as the prediction results. There are two factors that may affect the accuracy of the output results. One is the *prediction distance in time*, which is denoted as δ. Intuitively, the larger the δ, the less accurate the prediction will be. Another factor that may have an impact on prediction accuracy is the value of τ, which is the length of each timeslot when training and testing the model and in this paper, we refer to it as *prediction granularity*.

To explore the influence of these two factors on the prediction accuracy, we prepare the following two measurements. Firstly, we keep the prediction granularity constant ($\tau = 1$ minute) and change the value of δ to see its impact on the prediction accuracy. Secondly, LSTM models with different prediction granularity are trained, and then δ is kept unchanged to see the impact of prediction granularity on accuracy.

We have two observations on the performance of using the LSTM prediction algorithm to predict future contacts for our scheduling system.

1) Far-future predictions are less accurate: Firstly, we fix the prediction granularity as 1 minute, and under this granularity, we train our LSTM model with $H = L = 30$ timeslots. We predict the coordinates of all vehicles for the next 30 timeslots at each timeslot, that is, δ ranges from 1 to 30. After that, the recall and precision rates corresponding to different δ in the testing period (a total of 2 hours) are counted to explore the impact of δ on prediction accuracy.

Figure 2 shows the precision rate and recall rate of the δ-timeslot-ahead predictions. We can see that the precision rate and recall rate are always less than 1, which means there are always incorrect and unpredicted contacts. As δ increases, both the precision rate and recall rate decrease monotonically, which means we should be more careful to exploit prediction results for far-future timeslots. These prediction results are expected to include hints of future contacts, which should be valuable for making optimal scheduling decisions, but the high probability of incorrect predictions may make these predictions less valuable even harmful.

2) Prediction granularity has little influence on the prediction accuracy: We have already known that far-future predictions are less accurate, however, if the δ is the same, it is still unknown whether there will be a significant difference for the accuracy obtained by using different prediction granularity. Therefore, we train three LSTM models with a prediction granularity of 1min, 2min, and 4min respectively. We denote these three models as LSTM-1, LSTM-2, and LSTM-4. Then

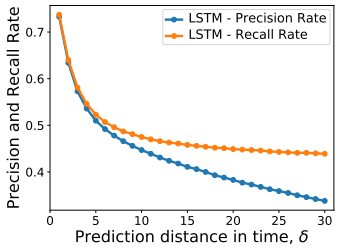


Fig. 2: With the increase of prediction distance in time (δ), the precision rate and recall rate decrease monotonously.

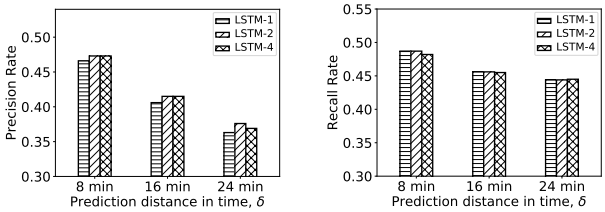


Fig. 3: The precision and recall changes caused by the predicted granularity are smaller than δ.

we select three fixed δ (8min, 16min, and 24min), and in each timeslot t, we use the above three models to predict the coordinate of all vehicles at future three timeslots (t+8, t+16, $t + 24$).

As we can see from Figure 3, the performance of the three models are approximately equal in both the prediction precision rate and recall rate. Compared with the difference brought by different δ, the precision and recall changes caused by the predicted granularity are very small when under the same δ. This feature helps us better realize the loosely coupled relationship between the scheduling system and the prediction algorithm, which will enhance the practicability of our scheduling scheme. Because the prediction granularity cannot be changed after the model is trained, if the impact of the prediction granularity is great, we must carefully design a prediction granularity when model training in order to obtain better results. If so, the scheduling system will rely on certain specific trajectory prediction algorithms. But now we can only use the output of the prediction algorithm and no longer care about its specific implementation.

V. System Design

In this section, we design a scheduling system to offload elastic traffic under an imperfect trajectory prediction environment, and the system should make the scheduling decisions in an online manner.

A. A Simple Case: scheduling a single request in an opportunistic network

As we explained before, a vehicle can communicate with one RSU/vehicle when it moves into the transmission range of the RSU/vehicle, and the duration of each communication contact is limited. Therefore, a transmission path involves two dimensions, *i.e.*, space and time. Finding paths for a request in time-varying opportunistic networks is usually solved using Time Expansion Graph (TEG).

Assume $\mathbb{V}_1$ requests a file at t_0, and the deadline of this request is t_4. There are other two vehicles in the scene, $\mathbb{V}_2$ and $\mathbb{V}_3$. Figure 4 shows the TEG to make a scheduling decision (*i.e.*, determining transmission paths) for this request. In the TEG, the content server is represented by the node S, and all RSUs are represented by nodes $\mathbb{U}_j(j = 1, 2, ...)$. S is connected to each $\mathbb{U}_j$ because they can communicate anytime they want. For each vehicle $\mathbb{V}_k$ in the system, we create a set of nodes $\mathbb{V}_{k,i}$, and $\mathbb{V}_{k,i}$ represents the vehicle $\mathbb{V}_k$ during the timeslot t_i.

The edge connecting $\mathbb{V}_{k,i}$ with $\mathbb{V}_{k,i+1}$ (*i.e.* the same vehicle in consecutive timeslots) means the vehicle can carry the transmitted content across timeslots. This kind of edge always exists for any k and i. The edge connecting $\mathbb{V}_{k,i}$ with $\mathbb{V}_{k\prime,i}$ (*i.e.* two vehicles in the same timeslot) means $\mathbb{V}_k$ can transmit content to $\mathbb{V}_{k\prime}$ in the timeslot t_i. This edge exists when and only when the prediction algorithms predict there will be a contact between the two vehicles $\mathbb{V}_k$ and $\mathbb{V}_{k\prime}$ in the timeslot t_i.

As described above, at t_0 we can construct the TEG based on the prediction results given by a prediction algorithm, as shown in Figure 4. Finding usable paths for the request of $\mathbb{V}_1$ is equal to finding paths between S and $\mathbb{V}_{1,4}$ in the graph, which is a traditional problem and has been solved. In our example, there are four usable paths, which are presented at the bottom of the graph.

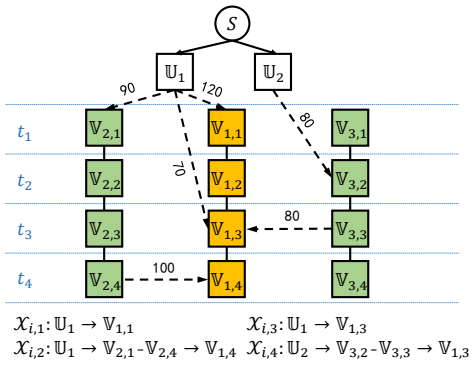


Fig. 4: An example of Time Expansion Graph (for a single request initiated by $\mathbb{V}_1$).

In our system, the TEG is constructed when the system starts to run. In the beginning, the TEG only contains node S and each $\mathbb{U}_j$. Then, all vehicles that participate in this offloading system will regularly report their historical coordinates. The trajectory prediction method will obtain their near future coordinates, which can be used to create the nodes $\mathbb{V}_{k,i}$ in the TEG. If a vehicle sends out a request, in addition to the request information, it will also send the latest historical coordinates of the recent L timeslots, which will be used by the trajectory prediction method to update this target's future coordinates.

B. Challenges in designing a scheduling system

The above example considers only a single request. The TEG is constructed for four timeslots (t_1 to t_4) because the deadline of the request is t_4. The TEG graph covers the whole lifetime of this request. However, the lifetime of a request can be very long, and there can be many requests, and they arrive at different timeslots with different deadlines.

Let us name the time covered by one scheduling operation as a *scheduling window*. For any request, the TEG always covers the whole lifetime, but the scheduling window is adjustable. The system should carefully determine the length of a scheduling window. Taking the whole lifetime as a scheduling window brings at least two problems. First, the time to find usable paths between two nodes is at least $O(V^3)$, where V is the number of nodes in the graph, which means a larger graph can make it more difficult to make scheduling decisions in a real-time manner. Second, a large scheduling window means that predictions for far-future timeslots are used during this scheduling operation. As we demonstrated in Section IV, far-future predictions are more likely to be incorrect, and scheduling decisions based on incorrect predictions will fail and degrade the throughput of the vehicle network. A short scheduling window also has a negative effect. Paths across more timeslots than the scheduling window cannot be used by the scheduling system, which may also deteriorate the system performance. Therefore, **the size of each scheduling window, denoted by λ, should be carefully determined**. In this paper, for request i, we refer to the ratio of its remaining demand to the remaining lifetime as its *demand intensity*, denoted by D_i. The determination of the scheduling window should first adapt to the trajectory prediction algorithm used by the system. Then, the scheduling window of different requests should adapt to their respective request intensity.

The other problem is how to schedule multiple requests. Usable paths of different requests may compete for the same networking resource, and we have to decide which request can use the resource. Also, there may exist multiple usable paths for a request. Therefore, **how to scheduling multiple requests with multiple usable paths also needs to be decided**. It is worth noting that the competition of multiple requests and multiple paths is not a new problem in this field, and there are already some methods [8], [26]. The key difference between our method and the existing methods is that we give the nearest useable path the highest priority, which is inspired by our measurement results.

We solve these two problems, *i.e.*, determining the scheduling window size and solving the resource competition among requests, in V-C and V-D respectively.

C. Determine scheduling window size and reduce the impact of imperfect predictions

After a request arrives, the system first needs to determine the scheduling window for this request. If the scheduling window is smaller than its remaining lifetime, the system continues to determine a new scheduling window on the last timeslot of the current scheduling window. This process will be repeated until the request is satisfied or expires. That is, a request is satisfied through multiple scheduling windows. Next, we will describe how to determine the scheduling window's size, which is a two-step approach.

1) determining the maximum scheduling window: The original intention of adopting a small scheduling window instead

of one entire lifetime comes from the fact that far-future predictions are less reliable. Therefore, a straightforward method is to select a fixed value as the size of all scheduling windows based on experience. This approach is simple, but it is difficult to apply it to scenarios with different prediction algorithms. Different cities may use different prediction algorithms. Besides, a model's prediction accuracy may vary for some reasons, such as periodic retraining of the model. Therefore, the way to determine the scheduling window should be adaptive to different prediction accuracy.

Specifically, after the system starts running, we can collect the scheduling decision failure rate, which is defined as the ratio of the number of failed decisions to the total number of decisions. This is easy to do since RSUs can report to the central controller whether RSU-to-vehicle transmissions occur or not. The occurrence of contacts between vehicles can be reported by target vehicles. This communication burden is not significant because target vehicles can report the cumulative results over a while instead of reporting them in real-time. When collecting, we can count the proportion of transmission failures corresponding to different δ. We update these statistical results regularly and regard the reason for the failure as a prediction error. We can know from the measurement in Section IV that the error rate will increase monotonically with the increase of δ. We can set a maximum error rate threshold η. According to η and statistical results, we can find the maximum prediction time distance λ^M whose error rate does not exceed the threshold η. When the system is initialized, we can set a fixed value as λ to run the system.

If λ^M is used as the scheduling window size, the prediction error rate would be within a range that we can accept. However, this step alone is not enough because all requests using the same window are prone to long processing queue delays. For example, the λ^M value is 5 timeslots, and we use this value as λ for all requests. We assume that the scheduler can handle up to 150 requests without significant queuing delay. 100 new requests arrive at t_0. The first scheduling window size begins at t_0, while the second begins at t_5. However, 75 new requests arrive at t_5, so the total number of requests that need to be processed at t_5 is 100+75=175, which will cause an obvious queuing delay. As can be seen from the above example, not only the new requests but also the old requests that arrived λ^M timeslot ago should be processed by the scheduler. If there are a lot of new requests arriving at both two timeslots, significant queuing delays will be introduced as the scheduler can't process the requests quickly. Therefore, λ_M cannot be directly used as the scheduling window size for all requests, and the following step needs to be combined.

2) assigning different window sizes for requests based on their demand intensity: As mentioned above, different requests have different demand intensity D_i. Here, we take the demand intensity as the basis for assigning different window sizes for requests. A small scheduling window means that the system may lose some cross-window V2V paths, but this has two benefits. First, the scheduling complexity in a single window is lower because fewer future timeslots are covered. More importantly, it means that this request has more scheduling opportunities. Since the trajectory prediction algorithm will constantly update the TEG contact information, the paths used by the scheduling with smaller windows are more reliable. However, this also means more frequent scheduling, which increases the processing burden of the system. Therefore, we assign small scheduling windows to high-intensity requests. Specifically, for multiple requests that arrive at the same time, the one with the highest intensity is assigned with the minimum scheduling window size λ_{min}, usually $\lambda_{min} = 2$ timeslots, while the one with the lowest intensity is assigned with the maximum scheduling window size λ_{max}, which equals to λ^M. The scheduling window size for request i is calculated as:

$$\lambda_i = \lambda^M - \frac{(D_i - D_{min})(\lambda^M - 2)}{(D_{max} - D_{min})} \tag{7}$$

where D_i is the demand intensity of request i, while D_{max} and D_{min} represent the maximum and minimum demand intensity among all requests at the current timeslot, respectively.

In summary, we determine the size of the scheduling window for each request in this two-step approach. We first use an adaptive method to determine the system's maximum scheduling window size, that is, the value of λ^M. Then we use the demand intensity to determine the window size for each request.

D. Scheduling for multiple requests with multiple paths

After the scheduling window for request i is determined, the system should make scheduling decisions within the scheduling window. Since the request may be satisfied with multiple scheduling windows, the demand of content that should be scheduled in a single window, which is termed as *sub-demand* q_i, is also a part of the total residual demand. The proportion of q_i in Q_i is equal to the ratio of the window size to the remaining lifetime, *i.e.*

$$q_i = \lambda_i \frac{Q_i^{'}}{td_i - t_{now}} \tag{8}$$

where $Q_i^{'}$ is the remaining demand and t_{now} is the current timeslot.

The problem that we should address is how to make scheduling decisions. There are already some methods to solve this problem, such as the MMCD method [8]. The key idea of this approach is as follows. MMCD sorts multiple requests according to their deadline and gives higher priority to the requests closer to the deadline to solve the competition among multiple requests. There may exist multiple useable paths for a single request. These usable paths are ranked from high to low according to their capacity. The scheduler selects these paths gradually to make scheduling decisions until the request is satisfied or all paths are used. Their experiments show that MMCD can achieve good throughput and delay. These results can only be achieved under perfect trajectory prediction algorithms as they only use path capacity to choose candidate paths. A contact's capacity is related to the distance of the two nodes, and when trajectory prediction is not accurate, the capacity information is not reliable. To solve this defect, we propose ***C**losest **P**ath with **M**aximum **C**apacity first (CPMC)*.

We still use EDF (The Earliest Deadline First) to resolve the contention among multiple requests. The difference is the way to make scheduling decisions on multiple transmission paths. In this paper, we call these usable transmission paths as *candidate paths*. We observe that each candidate path has two properties. One is the ***path capacity***. The capacity of a V2I path is equal to the capacity of RSU→ target link, while the capacity of a hybrid path is the smaller value of the capacity between the two opportunistic links. The second property is the ***last-hop time***. It refers to the timeslot where the last hop transmission occurs on the path. In a V2I path, it refers to the timeslot at which the RSU→target occurs, while in a hybrid path, it is the timeslot at which the helper→target occurs.

Let us illustrate our CPMC strategy with the example in Figure 4. The capacity and last-hop time of the four candidate paths in Figure 4 are 120, 90, 70, 80 and t_1, t_4, t_3, t_3. We think that the last-hop time is more important regarding these two properties, which is inspired by our measurement result in Section IV. Since last-hop time indicates the farthest prediction, the smaller the last-hop time's value, the more reliable the prediction results will be. In other words, the path is more likely to exist in the future. Combining these two properties, we propose the principle of path selection, that is, the closest path with maximum capacity first. We first choose the path with the smallest last-hop time value, and for multiple paths with the same last-hop time, we prefer the one with the maximum capacity. As a result, for the four candidate paths in this example, the order we choose is r_1, r_4, r_3, and r_2.

Algorithm 1 Scheduler (CPMC)

Input: sub-demand q_i, current time τ, window size λ_i
Output: scheduling decisions: Ω

for each t in range $(\tau+1, \tau+\lambda_i)$ **do**
  $M \leftarrow$get candidate path for i whose last-hop time is t
  Rank M according to path capacity
  for each path r in M **do**
    if r is hybrid path $(\mathbb{U}_j \xrightarrow{t_1} \mathbb{V}_k \xrightarrow{t} \mathbb{V}_i)$ **then**
      $X \leftarrow min(\mathcal{B}^{t_1}_{\mathbb{U}_j}, \mathcal{C}^{t_1}_{\mathbb{U}_j,\mathbb{V}_k}, \mathcal{C}^{t}_{\mathbb{V}_k,\mathbb{V}_i}, q_i)$
      if $X > 0$ **then**
        add $(i, \mathbb{U}_j \xrightarrow{t_1} \mathbb{V}_k \xrightarrow{t} \mathbb{V}_i, X)$ to Ω
        Update $\mathcal{B}^{t_1}_{\mathbb{U}_j}, \mathcal{C}^{t_1}_{\mathbb{U}_j,\mathbb{V}_k}, \mathcal{C}^{t}_{\mathbb{V}_k,\mathbb{V}_i}$
    if r is V2I path $(\mathbb{U}_j \xrightarrow{t} \mathbb{V}_i)$ **then**
      $X \leftarrow min(\mathcal{B}^{t}_{\mathbb{U}_j}, \mathcal{C}^{t}_{\mathbb{U}_j,\mathbb{V}_i}, q_i)$
      if $X > 0$ **then**
        add $(i, \mathbb{U}_j \xrightarrow{t} \mathbb{V}_i, X)$ to Ω
        Update $\mathcal{B}^{t}_{\mathbb{U}_j}, \mathcal{C}^{t}_{\mathbb{U}_j,\mathbb{V}_i}$
    $q_i \leftarrow q_i - X$
    if $q_i \leq 0$ **then**
      return Ω
return Ω

In summary, we first use EDF to resolve the contention among multiple requests. Then, Algorithm 1 is used by the i-Offload system to make scheduling decisions for each request i. The algorithm's input includes the sub-demand q_i, current timeslot τ, and the scheduling window size λ_i. The output is a series of scheduling decisions, represented as set Ω. We do the following operations for each timeslot t from near to far in the scheduling window. First, we get all candidate paths whose last-hop time property is t for request i, that is, all paths from node S to $\mathbb{V}_{i,t}$ in the TEG. We put all candidate paths into the set M and sort the paths according to their capacities from high to low. Then we make a scheduling decision on each path r. We first determine the scheduling amount X, which is the minimum value among the remaining bandwidth of the RSU $\mathcal{B}^{t_1}_{\mathbb{U}_j}$, the path capacities, and q_i. If X is greater than 0, then add the scheduling decision, represented by a triplet (request-id, transmission path, transmission amount), to the set Ω. After that, the remaining bandwidth of the RSU and link capacities will be updated. Finally, if $q_i - X$ is greater than 0, it means that the current scheduling demand has been satisfied. Otherwise, continue to the next candidate path until all candidate paths have been executed.

The time complexity of the entire algorithm is $O(\lambda L)$. The value of λ will not be very large, usually small than 100, and L is the size of set M. Since each vehicle's communication range is several hundred meters, the number of other vehicles located in this range is limited. Therefore, the value of λL is on the order of 10^3.

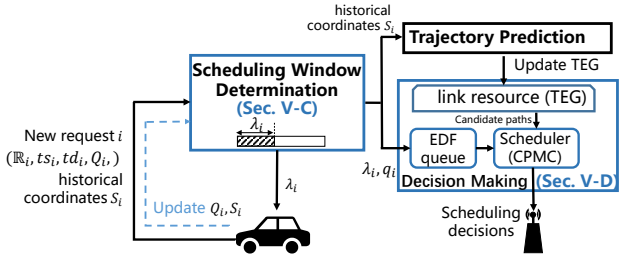


Fig. 5: System Overview.

E. i-Offlaod system

Based on the above designs, we propose our scheduling system, *i.e.* **i-offload**, and the overview is shown in Figure 5. When new request $i = (\mathbb{R}_i, ts_i, td_i, Q_i)$ and the target's historical coordinates $\mathcal{S}_i$ arrive at timeslot ts_i, the first module it should go through is ***Schedule Window Determination (SWD)***. This module will determine the scheduling window's size for this request and form a sub-request from request i, which will be sent to the following modules. Then, the SWD module will inform the target vehicle $\mathbb{R}_i$ that the current scheduling window size is λ_i. As long as the request is alive or not be satisfied, the module will treat it as an *active request*. At the last timeslot of the current scheduling window, the target vehicle will send an update message about the remaining demand Q_i and the latest historical trajectory S_i. In this way, the system can obtain more accurate prediction results and request demand information. The system can then determine a new scheduling window and continue to make decisions for it until the request times out or is satisfied.

The system uses the ***Trajectory Prediction*** module and ***Decision Making*** module to make scheduling decisions for the sub-request. The Trajectory Prediction module is a third-party application, and its specific implementation is not included in

our design. The Decision Making module contains two components, *i.e.*, an EDF queue and a scheduler which implements Algorithm 1. After all requests in the EDF queue are processed according to Algorithm 1, if there still exist free transmission links in the next timeslot, the scheduler will make further use of these remaining resources. Specifically, the system will make additional scheduling decisions with these free links for all active requests according to the CPMC principles.

The Trajectory Prediction module's output is used to update link resource information, and the time required to make prediction results will directly affect the delay in decision making. So we must evaluate the time required for existing trajectory prediction methods to produce prediction results. We choose the LSTM prediction method to predict the trajectory coordinates of 10,000 vehicles in the next 20 timeslots and record the required time. We conduct our measurements on an Intel Xeon CPU and an NVIDIA TITAN XP GPU. We repeat the measurement 1,000 times and take the average of the required time as our measurement result. The final result shows that it takes about 11 ms to get the coordinates of 10,000 vehicles, and the GPU utilization rate is 90%. And we can further reduce the time required for large-scale prediction by increasing the GPU's computing power. Therefore, the time required for existing trajectory prediction methods will not cause significant delay to our scheduling algorithm, and our system can be applied to large-scale urban scenarios.

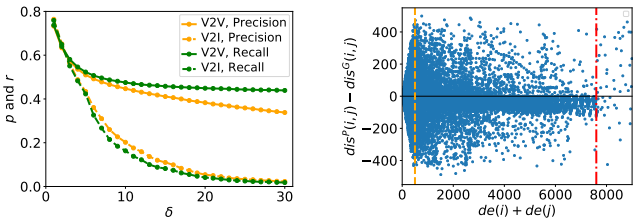
VI. Simulating Prediction Results with Different Accuracy

To evaluate our scheduling system's performance, we need to run it in a series of trajectory prediction algorithms with different accuracy. Therefore, in this section, we will discuss how to simulate prediction results with different accuracy. We first introduce an existing model, *i.e. fog-of-war*, and point out some shortcomings of this model. Based on the fog-of-war, we make some improvements and propose our model, *dense-fog*.

A. The fog-of-war model

The fog-of-war model is proposed by Malandrino *et al.* in [7]. It assumes that it has had a ground-truth dataset of vehicles' locations within a period, which means future contacts have been known accurately. Based on this dataset, the fog-of-war model adds noises to real contacts to generate imperfect prediction results. The noises are generated using parameters specified by users to simulate prediction algorithms with different performance characteristics. Specifically, for each real contact in the dataset, do the following two steps to generate the corresponding simulated result.

1) A Gaussian distribution is generated to represent the prediction noise, where the mean is zero, and the variance is σ_p^2. The value of σ_p^2 depends on the type of this contact. If it is a V2I contact, $\sigma_p^2 = \sigma_{0,p}^2 \Delta t$, where $\sigma_{0,p}$ is the accuracy parameter specified by users and Δt is the prediction distance in time. If the contact is V2V type, $\sigma_p^2 = 2 * \sigma_{0,p}^2 \Delta t$.
2) Extract a realization v from the above distribution. If $|v| \leq 1$, the model thinks that this contact can be predicted accurately. Otherwise, the contact is evicted, and a new, spurious contact is created, and the contact nodes are chosen among all nodes randomly.



(a) (b)

Fig. 6: (a). Precision rate and recall rate for V2V and V2I contacts. (b). $de(i)$ represents the deviation between the prediction position and its real position for vehicle $\mathbb{V}_i$. $dis^P(i,j)$ and $dis^G(i,j)$ represents the distance between vehicle $\mathbb{V}_i$ and vehicle $\mathbb{V}_j$ in the prediction results and in the ground truth, respectively.

This model can generate multiple sets of prediction contacts with different accuracy by changing the value of $\sigma_{0,p}$.

However, we find that the fog-of-war model is not sufficient to characterize the LSTM-based prediction algorithm.

Firstly, this model can only characterize prediction algorithms with the same precision and recall rate. However, these two metrics are usually unequal, which can be illustrated by Figure 2. Let's think back the definition of precision rate ($N_{tp}^{\delta}/N_p^{\delta}$) and recall rate ($N_{tp}^{\delta}/N_g^{\delta}$). When the denominators of these two metrics are the same, the two metrics always remain the same. Since the fog-of-war generates the prediction result one-to-one based on real contacts, the number of real contacts and the number of predicted contacts generated are exactly the same, *i.e.,* δ_p are always the same as δ_g.

Secondly, this model does not consider the particularity of stationary vehicles. In practice, the contacts between two long-time stationary vehicles can be predicted accurately by the LSTM method.

Thirdly, the assumption about the prediction inaccuracy of V2V contacts and V2I contacts may not hold in the LSTM method. The fog-of-war model assumes that the prediction inaccuracy of V2V contacts is twice that of V2I contacts, which is reflected in their variance calculation formula. They argue that the two communicating entities of a V2V contact are both moving, while only one entity of a V2I contact is moving. However, this assumption does not apply to the previous LSTM method. To reveal this, we take the following measurements.

We first measure the prediction and recall rate of the LSTM method for V2I and V2V contacts. The results are shown in Figure 6-(a), which demonstrates the assumption in the fog-of-war model does not hold in our scenario. We conjecture that it is because here we concern about whether a contact can occur (*i.e.*, the distance between two vehicles) instead of the accuracy of predicted locations of each individual vehicle. For example, if two vehicles have the same recent historical trajectories, the LSTM tends to make the same incorrect predictions for two vehicles. Although the predicted locations are incorrect, the algorithm still predicts there will be a contact, which is a correct prediction. To verify the

conjecture, for each contact (assuming it is between $\mathbb{V}_i$ and $\mathbb{V}_j$), we further plot $de(i) + de(j)$ (the sum of the deviation between the predicted location and the real location of each vehicle) and $dis^P(i,j) - dis^G(i,j)$ (the deviation between the predicted distance between two vehicles and their real distance) in Figure 6-(b). Here, each blue dot represents the information about a contact. We can see that only a small part of the points have abscissa values less than 500m (the left part of the yellow dashed line). This indicates that a lot of predicted contacts will really occur, although the predicted locations of the corresponding vehicles are incorrect.

B. The dense-fog model

Based on the above measurements and analyses, we make some improvements on the fog-of-war and propose our model, *dense-fog*. The dense-fog model also uses real contacts as ground truth and generates prediction results with different accuracy by making the following changes on the fog-of-war.

1) Find out stationary vehicles. Contacts between these stationary vehicles can be predicted correctly.
2) For other contacts, we continue to use the fog-of-war to generate simulated contacts. The difference is that when calculating σ_p^2, both V2V and V2I contacts are calculated by $\sigma_p^2 = \sigma_{0,p}^2 \Delta t$.
3) Make up for the defect that the number of contacts generated by the simulation is different from the ground truth by adding or discarding some contacts generated in the second step. Specifically, we find that the growth rate of the number of prediction contacts is proportional to δ, which can be seen in Figure 7-(a). We use the parameter μ to represent the proportional, and the number of contacts that need to be added or deleted is calculated as $N_m = \mu\delta$. When the value of N_m is positive, we randomly add N_m contacts which are not generated in step 2 and will not occur actually. Otherwise, delete N_m contacts randomly from contacts generated in step 2.

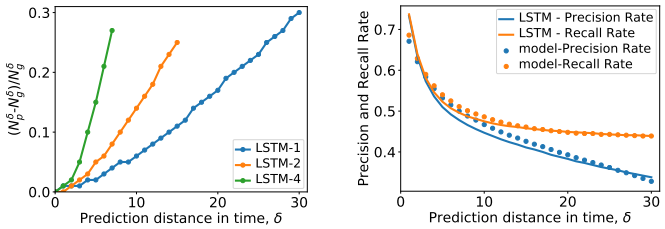
By changing $\sigma_{0,p}$ and μ, our dense-fog model can generate a series of simulated contacts with different accuracy. For example, the prediction results generated by the LSTM prediction algorithm trained in Section IV can be simulated by the dense-fog model with $\sigma_{0,p} = 1.04$ and $\mu = 0.001$ (Figure 7-(b)). The dots in this figure represent the contacts generated by our dense-fog model. It can be seen from the figure that both the precision rate and recall rate are very close to the results generated by the LSTM method, which shows the feasibility of our model.

VII. Evaluation

In this section, we will evaluate our *i-Offload* system.

A. Settings

Road Topology and Vehicle Trajectory. Koln dataset proposed in [34] is used as the vehicle trajectories in our evaluations. This dataset covers a region of 400 km^2 for 24 hours in a working day. In this paper, we choose two hours (08:00-09:59) as the running time of our system, and during this period, there are 44,000 vehicles. The timeslot size of the system is set to 10s. We take vehicle trajectories in the Koln dataset as ground truth and use the dense-fog model proposed in Section VI to generate prediction results with different precision rates and recall rates.



(a) (b)

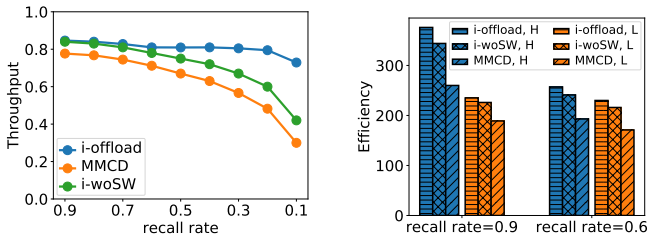
Fig. 7: (a). The growth rate of the number of prediction contacts is proportional to δ. (b). Precision rate and recall rate for the simulation contacts with $\sigma_{0,p} = 1.04$, $\mu = 0.001$ and the real LSTM prediction contacts.

Network and Communication. Similar to [7], we assume that the deployment density of RSUs is $1/km^2$, with a total of 400 RSUs. These RSUs are deployed in 400 hot-spots, which are obtained by K-Means clustering of vehicle trajectories. The bandwidth from each RSU to vehicles is set to 200Mb per timeslot. According to [35], high throughput and reliable connection can be maintained within a distance of 200 meters. Therefore, we set the communication range radius of all nodes to 200m. Referring to the experimental results in [35], we set the mapping relationship between contact capacity and contact distance as follows ([m, m]: Mbps): {[0, 40]: 23; [40, 80]: 21; [80, 120]: 14; [120, 160]: 8.5; [160, 200]: 4}. For example, the contact capacity of two vehicles is set to 23 Mbps if the distance is smaller than 40 meters.

Content and Request. The size of the requested content is a random value from 100MB to 300MB. The content is divided and encoded into several blocks before transmission, and the size of each block is 128 kbit. In this paper, we denote the proportion of target vehicles in the total vehicles as tp, and the value of tp is 0.5 in our evaluation. These target vehicles generate a total of 20,842 requests randomly within two hours, and the lifetime of each request ranges from 8min to 20min. To assess the network burden introduced by these requests, we calculate the ratio of all requests' demand to the sum of contacts capacities between target vehicles and RSUs, and the result is 1.24. This value means that even if the prediction is 100% accurate, the transmission task could not be completed by V2I communication alone.

Baseline and Metrics. We compare our i-Offload system with two methods. One is MMCD [8]. MMCD first sorts the requests according to their deadline and then prioritizes the transmission path with a large capacity when choosing the candidate path. The other method is the i-offload system without the Schedule Window Determination module, denoted as i-woSW. Unlike the i-Offload, the scheduling window size of MMCD and i-woSW is the minimum between the request's lifetime and the maximum prediction window.

We use the following two metrics To evaluate these three methods. The first is the ***scheduling throughput***, which refers



(a) scheduling throughput (b) scheduling efficiency

Fig. 8: Comparison of scheduling performance under different prediction recall rates.

to the ratio of the amount of data offloaded through the vehicular network to the total demands. The second is the ***scheduling efficiency***, which is defined as the average amount of data that each scheduling decision can offload, that is, the ratio of the total offloading amount to the number of scheduling decisions.

Urban Deployment Scenario. To evaluate our system deeply, we design two simulation scenarios. The first is ***Well-deployment Scenario***, in which the vehicular network deployment has been finished. In this scenario, there are more RSUs, and these RSUs are deployed in a hot-spot pattern. This deployment pattern is beneficial for improving scheduling throughput [36]. Besides, the Vehicle-to-Vehicle communication and cooperation platform has also achieved full coverage, which means that all vehicles can and are willing to become helpers. The other scenario is the ***Early-deployment Scenario***, where the deployment of RSUs are sparse or non-hotspot deployed, and the V2V cooperation platforms are not widely deployed.

Next, we will evaluate our scheduling system using scheduling throughput and efficiency under the two scenarios.

B. Performance in Well-deployment Scenario

In this subsection, we evaluate the offloading throughput and efficiency of our method under the Well-deployment Scenario.

1) Throughput: We first evaluate the scheduling throughput. When the recall rate is constant, the throughputs under different precision rates are approximately equal. This is unexpected but reasonable since at the same recall rate but different precision rates, the number of contacts that are predicted correctly is always the same, and all three methods will make full use of these resources. Therefore, with the same recall rate but different precision rates, the system throughput is almost the same.

However, throughput performance varies significantly with different recall rates, as shown in Figure 8-(a). The abscissa in the figure represents the different prediction recall rates, from 0.9 to 0.1. The vertical axis represents the throughput. Three conclusions can be drawn from the figure. First, with the decline of the prediction recall rate, the throughput of all methods decreases, but the performance of the i-Offload is always better than MMCD and i-woSW. When the recall rate is 0.1, our method's performance improves by 42% compared with that of MMCD. Second, as the prediction recall rate declines, our algorithm's downward trend is slower, and a significant decline occurs when the recall rate is less than 0.2. Both of the above conclusions show that our method has a better performance in throughput when the trajectory prediction is imperfect. Third, the throughput of the i-Offload is always higher than i-woSW, which verifies our Scheduling Window Determination module's rationality.

2) Efficiency: Next, we evaluate the efficiency of offloading. We set the value of the recall rate as 0.9 and 0.6 to represent a high and a low recall rate, respectively, while the precision rate is set as 0.9 and 0.4 to represent different precision rates so that there are four prediction environments. Experimental results show that both precision rate and recall rate influence offloading efficiency. This result is shown in Figure 8-(b), where the blue bars represent the results under high precision rate, and the orange bars represent the results under the low precision rate, which are marked with 'H' and 'L' respectively in the figure.

From this figure, we can draw the following three conclusions. First of all, under the same recall rate, the prediction environment with a high precision rate has a high offloading efficiency, which is intuitive because the system always needs more scheduling decisions to satisfy requests at a low precision rate. Similarly, under the same precision rate and different recall rates, it is obvious that a high recall rate corresponds to high offloading efficiency. Secondly, no matter what kind of prediction environment, the scheduling efficiency of the i-Offload is 25%-45% higher than that of the MMCD method. Thirdly, by comparing i-Offload and i-woSW methods, it can be found that the existence of the Scheduling Window Determination module is also beneficial for improving scheduling efficiency.

3) Running time: To evaluate the practicability of the i-Offload system, we test our system's running time using different vehicles and contacts number. The number of vehicles ranges from 2,000 to 44,000. We conduct our measurement on an Intel Core i7-8700 CPU. As Figure 9 shows, the left coordinate axis represents the number of vehicles appearing in the scene, and the right coordinate axis represents the number of contacts. The vertical axis represents the average running time for i-Offload to make decisions. The average running time is the ratio of the total time that the system uses to make scheduling decisions to the number of all requests.

From the figure, we can see that the running time increases slowly with the number of vehicles and contacts. Even in the case of 44,000 vehicles and 2.5×10^7 contacts, our system can still make decisions within 160 ms, which shows that i-Offload can make decisions quickly.

C. Performance in Early-deployment Scenario

In this subsection, we evaluate the offloading throughput and efficiency under the Early-deployment Scenario. The deployment of vehicular networks includes two important parts. One is the deployment of RSUs, which contains the deployment pattern and density. The other is the deployment of Vehicle-to-Vehicle communication and cooperation platforms.

1) Infrastructure deployment: We denote the well-deployment scenario as 'h-400', that is, 400 RSUs are de-

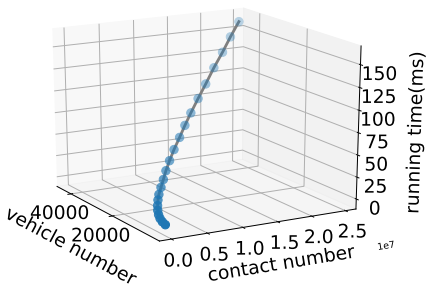
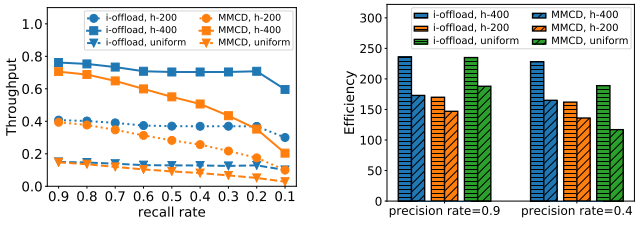


Fig. 9: The running time of the system.



(a) scheduling throughput (b) scheduling efficiency

Fig. 10: Scheduling performance under different RSU deployment scenarios.

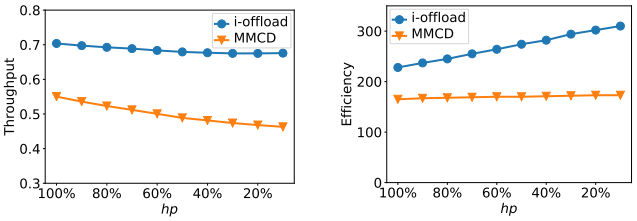
ployed in a hot-spot pattern. In the early stages of infrastructure deployment, we envision three possible scenarios.

1) h-200: There are only 200 RSUs in the scenario, and all of the RSUs are deployed in a hot-spot pattern.
2) uniform: There are 400 RSUs in the scenario, but all RSUs are deployed with a 'uniform' pattern, that is, urban areas are divided into equal sub-areas, and RSUs are deployed beside roads in the middle of each sub-area.

We first compare the offloading throughput in three deployment scenarios. The result is shown in Figure 10-(a). It can be seen that under the two early-deployment scenarios, both scheduling methods perform poorly, especially in the case of 'uniform' deployment. Thus it can be seen that the RSU deployment has a significant impact on offloading throughput. Secondly, in both early-deployment scenarios, i-Offload shows better performance. Besides, the significant performance degradation of i-Offload occurs only after the recall rate was less than 0.2, which shows strong robustness to different prediction environments.

Then, we compare the offloading efficiency under three deployment scenarios, and the results are shown in Figure 10-(b). From the figure, we can see that i-Offload has the highest efficiency in the three scenarios. Besides, comparing the blue bars with the orange bars in the figure, a low density of RSUs will lead to a low scheduling efficiency. This is because either target vehicles or helper vehicles need to get the data from the RSU, and a low RSU density will lead to a low upper limit of available data.

Finally, we find that the efficiency under the 'uniform' deployment pattern also has a high value. It is because there are few candidate paths available when making decisions, which leads to a low number of scheduling decisions. Therefore, based on the results of these two figures, our method, *i.e.* i-Offload, is better than MMCD in the early stage of infrastructure deployment.



(a) scheduling throughput (b) scheduling efficiency

Fig. 11: Scheduling performance under different hp.

2) Helper cooperation platform deployment: We assume that vehicles' cooperation is realized through a platform, and vehicles that install this platform can get data from other vehicles. Otherwise, vehicles can only obtain data from the infrastructure. In this paper, we denote the ratio of vehicles that install this platform to the total vehicle number as hp. An ideal vehicular network scenario is that all vehicles are equipped with this platform, that is, $hp = 1$. However, in the promotion process, it is impossible to achieve full coverage. Therefore, in this part, we keep the total number of vehicles in the urban area unchanged and change hp. In the experiment, the prediction results' recall rate and precision rate are set to 0.5 and 0.4, respectively, and the scheduling throughput and efficiency under different hp is evaluated.

The scheduling throughput and efficiency are shown in Figure 11. From Figure 11-(a), we can see that the scheduling throughput of i-Offload and MMCD both decreases as the hp decreases, but the decline in i-Offload is smaller. The following two reasons cause this. The first is that when V2V connections are reduced, more V2I links are used. This can be demonstrated by the offloading efficiency in Figure 11-(b), where when hp is decreasing, the scheduling efficiency of i-Offload is increasing. The second reason is that due to the Scheduling Window Determination module, cross-window paths are not used by the i-Offload to transfer data. Therefore, the reduction of hp has less impact on i-Offload than on MMCD.

In summary, our method is better than MMCD for the gradual deployment of vehicular networks under the imperfect prediction environment.

VIII. CONCLUSION

In this paper, we propose *i-Offload* to solve the elastic traffic offloading problem in vehicular networks under an imperfect trajectory prediction algorithm. First, we measure the LSTM prediction method that has been widely used recently, and obtain the factors that influence prediction accuracy. Then, based on the enlightenment from the measurement experiments, we design the i-Offload system to solve the above offloading problem. Finally, we point out the shortcomings of the existing method for modeling the accuracy of prediction results and make some improvements on it. The modified model can simulate a series of prediction results with specified accuracy, which can help evaluate scheduling algorithms'

performance. The experimental results show that our method achieves higher scheduling throughput and efficiency under an imperfect trajectory prediction environment.

REFERENCES

[1] Cisco, "Cisco annual internet report (2018-2023) white paper," https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

[2] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[3] H. Seo, K.-D. Lee, S. Yasukawa, Y. Peng, and P. Sartori, "Lte evolution for vehicle-to-everything services," *IEEE communications magazine*, vol. 54, no. 6, pp. 22–28, 2016.

[4] S. Yoon, D. T. Ha, H. Q. Ngo, and C. Qiao, "Mopads: A mobility profile aided file downloading service in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 5235–5246, 2009.

[5] K. Park, H. Kim, and S. Lee, "Reliable position based routing algorithm in vehicular ad-hoc network," in *2015 Seventh International Conference on Ubiquitous and Future Networks*. IEEE, 2015, pp. 344–349.

[6] P.-C. Cheng, K. C. Lee, M. Gerla, and J. Härri, "Geodtn+nav: Geographic dtn routing with navigator prediction for urban vehicular environments," *Mobile Networks and Applications*, vol. 15, no. 1, pp. 61–82, 2010.

[7] F. Malandrino, C. Casetti, C.-F. Chiasserini, and M. Fiore, "Content download in vehicular networks in presence of noisy mobility prediction," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1007–1021, 2013.

[8] K. Ota, M. Dong, S. Chang, and H. Zhu, "Mmcd: Cooperative downloading for highway vanets," *IEEE transactions on emerging topics in computing*, vol. 3, no. 1, pp. 34–43, 2014.

[9] G. Xie, A. Shangguan, R. Fei, W. Ji, W. Ma, and X. Hei, "Motion trajectory prediction based on a cnn-lstm sequential model," *Science China Information Sciences*, vol. 63, no. 11, pp. 1–21, 2020.

[10] S.-J. Qiao, K. Jin, N. Han, C.-J. Tang, and G. Gesangduoji, "Trajectory prediction algorithm based on gaussian mixture model," *Journal of software*, vol. 26, no. 5, pp. 1048–1063, 2015.

[11] S. Qiao, D. Shen, X. Wang, N. Han, and W. Zhu, "A self-adaptive parameter selection trajectory prediction approach via hidden markov models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 284–296, 2014.

[12] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 4363–4369.

[13] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 353–359.

[14] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2267–2272.

[15] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 399–404.

[16] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of network and computer applications*, vol. 37, pp. 380–392, 2014.

[17] Z. Li, Y. Liu, H. Zhu, and L. Sun, "Coff: Contact-duration-aware cellular traffic offloading over delay tolerant networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 5257–5268, 2014.

[18] Y. Li, D. Jin, Z. Wang, L. Zeng, and S. Chen, "Coding or not: Optimal mobile data offloading in opportunistic vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 318–333, 2013.

[19] X. Zhu, Y. Li, D. Jin, and J. Lu, "Contact-aware optimal resource allocation for mobile data offloading in opportunistic vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7384–7399, 2017.

[20] C.-M. Huang, M.-S. Chiang, D.-T. Dao, W.-L. Su, S. Xu, and H. Zhou, "V2v data offloading for cellular network based on the software defined network (sdn) inside mobile edge computing (mec) architecture," *IEEE Access*, vol. 6, pp. 17 741–17 755, 2018.

[21] F. Mezghani, R. Dhaou, M. Nogueira, and A.-L. Beylot, "Offloading cellular networks through v2v communications—how to select the seed-vehicles?" in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[22] M. Sathiamoorthy, A. G. Dimakis, B. Krishnamachari, and F. Bai, "Distributed storage codes reduce latency in vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 9, pp. 2016–2027, 2014.

[23] T.-Y. Yu, X. Zhu, H. Chen, and M. Maheswaran, "Hetcast: Cooperative data delivery on cellular and road side network," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–6.

[24] R. Ding, T. Wang, L. Song, Z. Han, and J. Wu, "Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2015, pp. 1207–1212.

[25] M. Li, Z. Yang, and W. Lou, "Codeon: Cooperative popular content distribution for vehicular networks using symbol level network coding," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 223–235, 2010.

[26] S. Yang, C. K. Yeo, and B. S. Lee, "Maxcd: Efficient multi-flow scheduling and cooperative downloading for improved highway drive-thru internet systems," *Computer Networks*, vol. 57, no. 8, pp. 1805–1820, 2013.

[27] N. Wang and J. Wu, "Opportunistic wifi offloading in a vehicular environment: Waiting or downloading now?" in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[28] L.-l. Wang, Z.-g. Chen, and J. Wu, "Vehicle trajectory prediction algorithm in vehicular network," *Wireless Networks*, vol. 25, no. 4, pp. 2143–2156, 2019.

[29] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[30] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[31] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8483–8492.

[32] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms," *IEEE Robotics and Automation Letters*, 2020.

[33] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 919–928.

[34] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2013.

[35] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal, "Vehicular opportunistic communication under the microscope," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, 2007, pp. 206–219.

[36] F. Malandrino, C. Casetti, C.-F. Chiasserini, and M. Fiore, "Content downloading in vehicular networks: What really matters," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 426–430.