Scalable Reactive Vehicle-to-Vehicle Congestion Avoidance Mechanism

Mevlut Turker Garip, Mehmet Emre Gursoy, Peter Reiher, Mario Gerla
Department of Computer Science, University of California Los Angeles, California, USA
{mtgarip, memregursoy, reiher, gerla}@cs.ucla.edu

***Abstract*—The increasing popularity and acceptance of VANETs will make the deployment of autonomous vehicles easier and faster since the VANET will reduce dependence on expensive sensors. Many useful applications will be possible with the usage of VANETs, which will improve the safety and quality of trips for the owners of these vehicles. One of these applications is the avoidance of traffic congestion by smart dynamic rerouting. For scalability, current cloud-based solutions, like Google Maps traffic, update congestion levels after a time interval rather than providing real-time measurements. In this paper, we introduce a vehicle-to-vehicle congestion avoidance mechanism, which detects real-time congestion levels and reroutes vehicles accordingly to minimize their trip times. Our system is highly distributed and is, therefore, not subjected to the limitations of centralized congestion mechanisms. We show via simulation that our system can significantly decrease the trip times of vehicles as well as the average car density on the map. Our proposed system, with its checkpoint and offline path generation approaches, is more responsive to local congestion level changes and computationally less complex for least-congested-route calculations than state-of-the-art congestion avoidance mechanisms.**

I. Introduction

The dream of autonomous, connected vehicles will soon become a reality. Vehicular ad hoc networks (VANETs) enable cars to communicate with each other over a wireless channel by sending packets directly to their neighbors within radio range. Vehicles can collect and share basic information such as speed and location. Peer-to-peer communication without centralized coordination can occur at high frequencies. In addition to inter-vehicular communication, VANETs can also integrate central units through the deployment of specialized infrastructure, commonly referred to as road-side units.

Advantages of VANETs are manifold. Since traffic safety is a pressing concern, much work has gone into utilizing VANETs to prevent traffic accidents. Reports claim that autonomous vehicles could save numerous lives and prevent more than 2 million injuries related to car accidents every year [1]. Other potential benefits of VANETs include toll payment and content delivery. Major car manufacturers have already started their preparations to gain a competitive edge in the autonomous car market [1]. Such cars can be expected to have VANETs integrated into them.

We will focus on one particular benefit of VANETs: congestion avoidance. Congestion is a serious problem for big cities with high populations [15]. Internet-based solutions such as Google Maps traffic are not necessarily adequate for immediate response to highly dynamic, fine-grained traffic and congestion control. Instead, VANET protocols and vehicle-to-vehicle dissemination of information can be used to achieve timely information exchange. Cars that are already in congested areas can recommend or advise against the use of certain roads, and this VANET-assisted communication may help nearby vehicles re-route themselves around sudden traffic jams. It is not feasible to achieve such instant impact via Internet-based solutions that, at best, act on the order of tens of minutes; rather than milliseconds.

In this paper, we propose an efficient and scalable distributed vehicle-to-vehicle (V2V) congestion avoidance mechanism. Unlike the other systems proposed in the field, our mechanism does not require a global knowledge of the congestion levels on the map, but rather only the congestion levels of the roads that the vehicle might take. The systems that use proactive approaches usually require vehicles to have almost full knowledge of all the roads on the map, which may often times be redundant. Furthermore, they come with an accumulated cost of storage, and re-computations and future dissemination of information (according to changes in congestion levels) may become painful. Our reactive approach can work under reasonable amount of memory, and in cases where limited and/or partial information is available. We address its network usage by implementing a request/reply technique rather than relying on periodic broadcasts. Our experimental evaluation shows that the system is very reliable: In any case it performs better than a completely offline shortest-path calculation (e.g., Dijkstra), and its benefits increase as the level of congestion increases.

The rest of this paper is organized as follows: In Section 2, we discuss related work on traffic congestion avoidance using VANETs. In Section 3, we give an overview of possible design considerations and elaborate on our choices. Section 4 describes the technical aspects and implementation details of our system. In Section 5, we examine its performance under different loads of traffic and validate its effectiveness using two metrics. In Section 6, we conclude by re-iterating the characteristics of our congestion avoidance mechanism and suggesting further improvements for future work.

II. Related Work

Traffic congestion is a serious problem for urban environments, and there is significant amount of research incorporating VANETs to address this problem. As shown in [8], dynamic route planning, according to real-time traffic information, is an effective way of reducing trip time. [8] demonstrates the viability of predicting future traffic by considering vehicle density alongside travel time. [19] reports on a simulation environment that implements VANET-enabled Intelligent Transportation Systems (ITS). Their results show

decrease in travel time and vehicle density in congested areas, when related congestion information is disseminated.

Quantifying congestion without complicated sensors is another effort-worthy task. Existing literature and standards assume access to basic sensors in a car: speedometer, GPS, on-board radar, etc. Information from these devices are part of BSM/CAM messages that are regularly exchanged as part of existing VANET protocols [10]. In [6], contents of beacons are used to measure levels of congestion. In particular, vehicle density and speed are used as indicators of congestion. An alternative approach is presented in [13], where vehicles monitor their speeds to detect congestion. Their measurements are aggregated into a "congestion parameter", which the authors propose to adaptively broadcast within WSMs in 802.11p and thus reduce network load comparing to periodic broadcasting.

Congestion avoidance mechanisms can be reviewed under two main categories: centralized and distributed. Existing deployed solutions (e.g., Google maps with traffic data, smartphone applications such as Waze) rely on central servers that collect and share information. These require 4G/LTE/Wi-Fi connections and do not make use of inter-vehicular communications. Similarly, a cloud-based route optimization technique is designed in [11]. Vehicles report traffic conditions to a main server, and can acquire a minimum-time path to their destination from the server. This optimal path is calculated by the server using a flow deviation algorithm. [9] is another centralized approach, in which vehicles upload their intended destinations and trajectories to a main server through access points. The server then tries to predict future traffic based on current traffic and vehicles' intended destinations.

There is also work on fully-distributed congestion avoidance systems that rely mainly on inter-vehicular communications. [18] demonstrates that these systems can be designed with minimal infrastructure costs but still achieve promising results. Their network model and assumptions, however, are not compliant with 802.11p. A state-of-the-art congestion avoidance mechanism is presented in [12]. Vehicles measure their travel time on each road and store it in their database. They selectively broadcast some of their measurements and measurements they hear from other vehicles. The selection process depends on the freshness of the measurements and their map coverage. Each vehicle also holds an aggregated summary of the whole map, built and updated using the measurements they generate and hear. Congestion avoidance decisions are made by each car according to their knowledge of congestion levels. However, the success of this approach would be highly dependent on its broadcast frequency. More insight on this will be given in the subsequent sections. In contrast, [7] suggests the use of a query-response protocol to collect congestion-related information. Such queries are stored and forwarded by nearby vehicles. They are optimized by the addition of cache structures and query expiration times. [16] studies peer-to-peer systems in which vehicles hear from others that are geographically ahead of them. The intuition is that leading vehicles (or ones that are traveling in the opposite direction) should have a better understanding regarding what this vehicle may expect. In case there is an upcoming traffic anomaly or congestion, the vehicle can be warned ahead of time and re-route itself accordingly. This offers a reasonable way to avoid abrupt traffic situations, but does not consider a healthy detection, quantification and dissemination of congestion information for more detailed navigation decisions.

III. Congestion Avoidance: Preliminaries

A. Discussion on Earlier Work and Design Decisions

In this paper we choose to create a distributed congestion avoidance scheme, rather than employing centralized servers and/or road-side infrastructure. Also, we consider a reactive mechanism (i.e., request/reply) rather than a system that relies on all-out periodic broadcasts.

There are several advantages of centralized congestion avoidance algorithms. First, a centralized server can have a more global knowledge of its realm, assuming it can receive and hold information originating from anywhere on the map. This may help deduce more accurate navigation decisions, especially over longer distances. Second, centralized methods do not require computation power and the map knowledge of the vehicles; it shares very little information with a vehicle (e.g., an optimal route to intended destination), as opposed to continuous reports of congestion levels.

There are also inherent problems with using centralized algorithms:

- Inability to achieve micro-management and immediate response. Current VANET technology supports delivery of vehicle-to-vehicle beacons that are sent every 0.1 seconds. Therefore, networked cars can be extremely fast in warning their surroundings regarding a blocked road, accident, traffic congestion, etc. However, it is not entirely feasible to have a central server receive messages via 4G with such high frequency.
- Additional infrastructure for communication, e.g., RSUs. Existing centralized methods that use VANETs make use of specialized infrastructure. The deployment of such units would be costly. Considering standard VANET transmission range, too many RSUs might have to be deployed to cover an entire urban area.
- Need for 4G/LTE connection so that either nodes directly connect to a main server for cloud approaches.
- Single point of failure. A malfunctioning RSU might mean an entire area is blacked out. These may motivate cyberattacks on VANETs, e.g., DoS on RSUs.

Most of the current VANET functionality (e.g., safety and collision avoidance warnings) depends on frequent broadcasts. Much work has gone into creating efficient network protocols that are particularly suitable for VANETs. A traffic congestion avoidance system such as ours needs to disseminate as much information as possible, while retaining a reasonable level of network overhead. Unlike our system, the success of others, that periodically broadcasts congestion-related information, is very dependent on its broadcast frequency. Higher broadcast frequency shares more information, but causes redundancy and network degradation. There are also cases in which continuous broadcasts may provide no benefit despite the overhead: (1) There are very few or no vehicles in the vicinity of the sender. Information could be broadcast without anyone demanding or potentially making use of it. (2) If numerous vehicles are sitting in a terrible traffic jam, they could flood nearby recipients with redundant messages.

We therefore use a request-reply based system. Instead of introducing a constant communication and computation

overhead, our system scale better by being reactive. When there are no vehicles within its transmission range, the system will mostly remain idle. When there are exceedingly many, it would issue a request only when it should, and its peers would respond only if they have relevant useful information.

B. Measuring Congestion: Choice of Congestion Metric

The ultimate goal of a congestion avoidance system is to minimize the trip times of vehicles, which requires an accurate and up-to-date representation of the current traffic congestion levels. To choose the best route, the routing algorithm must calculate and compare trip times for multiple possible routes from a starting point to a destination. One simple approach for measuring congestion is measuring the time needed to travel on each road. However, this metric could be too sensitive to irrelevant incidents. For example, a car might make a pit-stop on a road, which would cause its trip time to rise dramatically. Also, in a road with several lanes, the car might be stuck in a lane that is very slow whereas the most other lanes move much faster. Systems that use the travel time metric acknowledge such situations, and usually assume that they will not exist [12]. This motivated us to find another relevant, but less sensitive (i.e., less prone to extreme cases) metric.

We assume that every vehicle has (or can easily obtain) some basic information about the map, including the length of each road. Knowing that average speeds are inversely proportional to the time needed to travel, one can easily convert one to the other, given the length of a road. Hence we decided to use average speeds as our congestion metric, but with a catch: A car will sample and average its own speed together with other vehicles' speeds that are traveling on the same road (and direction, if the road is bi-directional). This computation is powered by the exchange of heartbeat beacon messages (BSM/CAM) and will be detailed in the next section.

IV. Proposed System

A. Message Types and Information Exchange

For the communication among all cars, we assume standard signal range of the 802.11p protocol, which is 100 meters [17]. In addition to the Basic Safety Messages (BSM) that are regularly sent to nearby vehicles (as explained in [10] and standardized in [5]), two types of messages are implemented in our system: congestion request and response.

1) Beacons / Basic Safety Messages: There are several fields within a BSM that are of interest to our application. First, the current speed of a vehicle is obviously crucial in the speed averaging process. Second, the position of the vehicle can be used to determine which road the vehicle is on (so that the recipient can figure out whether the message is from a car in the same road). Third, if the road is bidirectional then the angle of the vehicle would be used. Even though BSMs can be sent as frequent as every 0.1 seconds, one can perform sampling over the stream of incoming BSMs (i.e., choose a meaningful subset instead of evaluating all) if the computational power required by the speed averaging process is undesirably high. Creation of a sophisticated sampling algorithm is beyond the scope of this paper, yet some algorithms can be found in [20]. BSMs can be correlated with their senders via sender IDs, and for each sender we chose to sample BSMs every 2 seconds in our implementation on the simulator.

2) Congestion Request Messages: When a car approaches the end of a road and can choose from two or more routes, it will need congestion measurements from other cars in order to make the choice that will minimize its remaining trip time. Hence, it broadcasts a congestion request message to all nearby cars in its communication range to obtain this information. The content of this message is a list of **roads of interest**; these are the roads that form the candidate paths to destination. The car will choose its next path based on the responses it gets. In other words, these are the roads that will help a car differentiate between multiple paths and choose one over the other.

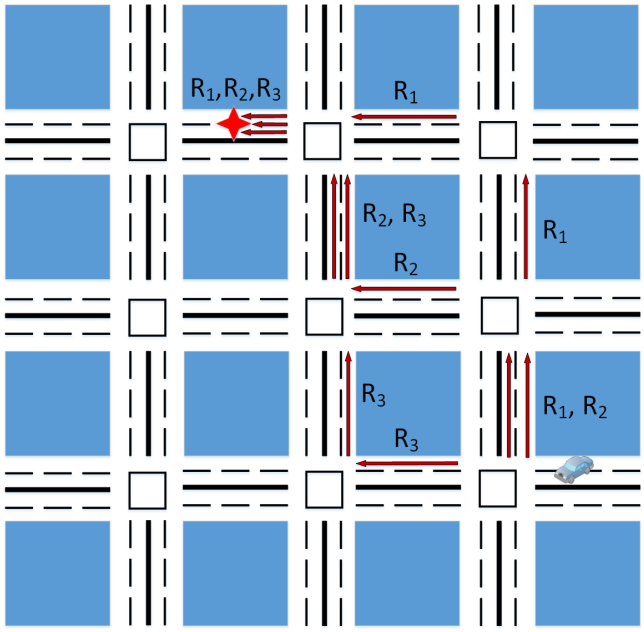


Fig. 1: An over-simplified routing decision with three potential route choices

Figure 1 demonstrates a scenario where the car is supposed to choose from R_1, R_2 and R_3 to arrive at its destination, denoted by a star. The arrows represent the roads of interest that the car would like to learn about. Note that there are two streets that are associated only with R_3. If R_3 was no longer a potential route for this car because one of these two streets were blocked, then the car would delete all additional roads due to R_3 from its list in congestion request, because they are no longer useful in choosing between its potential paths.

3) Congestion Response Messages: A congestion response message is sent only when a congestion request is received and there are related entries in the congestion information database of the receiving car, ensuring that no superfluous information is transmitted. The response includes congestion information about the roads of interest available at the receiver.

B. Congestion Information Database

1) Creating and Storing Congestion Information: In order to store and exchange congestion measurements, vehicles make use of *congestion info structs*. Each struct consists of the following fields:

Creator ID: The unique ID of the vehicle that created this measurement.

Edge ID: The unique ID of the road that this measurement belongs to. (Bidirectional roads are assumed to have two separate IDs, one for each direction.)

Average Speed: Average of the speed readings from all the cars on the same road with the car that crates this measurement.

Timestamp: Time of the measurement's creation to ensure the freshness of measurements and prioritize most recent ones.

2) Maintaining the Congestion Information Database: The congestion information database consists of the vehicle's own measurements and measurements learned from others as a result of the congestion request-response protocol.

Let m denote a measurement which is a congestion information struct. We define the following functions:

$c(m)$: unique ID of the car that created m
$e(m)$: unique edge ID of the road which m belongs to
$s(m)$: average speed in m
$t(m)$: timestamp of m

Let DB_x be an abbreviation for the local congestion database of car x. DB_x consists of zero or more congestion measurement entries. Let E be the set of all the edges (roads) in the map:

$$DB_x = \{m_1, m_2, m_3, \ \dots \ , m_n\}, \ 0 \leq n \leq |E|$$

There are two cases in which a measurement will be inserted to the database. The first is when the car creates its own measurement by sampling its own speed, together with the speeds of other cars on the same road it is passing along. In this case, the car will force this measurement into its database, overwriting an existing entry for that road (if one exists). The second is when measurements are obtained through congestion responses received from other vehicles. As stated earlier, a congestion response message CR consists of several measurements. Assume that a car sent out a congestion request with a set of interesting roads denoted by I. A congestion response message will be triggered at a recipient. The recipient will check whether any information regarding roads in I exists in its database. If not, no reply is sent. If a match is found, the response will have the following format:

$$CR = \{m_1, m_2, m_3, \ \dots \ , m_s\}, s \leq |I|$$

Each $m_i \in CR$ will then be given to Algorithm 1 as an input. The algorithm decides whether m_i should be dropped or added to a car's database.

t_α is a time threshold that determines when a measurement created by the car itself becomes outdated in its congestion information database. Cars prioritize the measurements that they created themselves over measurements originating from others. They will ignore fresher measurements heard from other cars if their own recordings are not at least t_α older than the received measurements.

At this stage, two important features should be highlighted. First, at all times, a car's database will contain at most one tuple for each road in the map. Second, since vehicles respond to congestion requests using the information in their database, if vehicle V_k gets measurement m_j from vehicle V_j in response to a request, V_k might later send m_j to a third vehicle V_s in response to the third vehicle's request. In other words, cars also disseminate congestion information learned from other cars in response to later requests. Since each measurement is associated with its creator by *Creator ID*, it can always be traced back to the car where it originated. Authenticity can be enforced using signatures [14].

Algorithm 1 Insertion of measurement from a CR message

```
if ∄ m ∈ DB_x, e(m) = e(m') then
    DB_x.add(m')
else
    m ← measurement from DB_x, where e(m) = e(m')
    if t(m') > t(m) then
        if c(m) ≠ c_x then
            DB_x.remove(m)
            DB_x.add(m')
        else if t(m') − t(m) > t_α then
            DB_x.remove(m)
            DB_x.add(m')
        else
            drop(m')
        end if
    else
        drop(m')
    end if
end if
```

C. Dynamic Rerouting

Our dynamic route planning technique helps cars avoid congestion by choosing the route with the minimum trip time. This optimal route is calculated using the congestion information available to the car. There are two limitations in real-time trip time optimization:

- Calculating a global optimum is challenging due to the amount of congestion information required. Additionally, the global optimum may change frequently since congestion levels are often highly dynamic. Therefore, we designed a checkpoint-based approach and minimize trip times for each sub-trip after dividing a long trip into several sub-trips.
- Running a link-state shortest-path algorithm (e.g., Dijkstra) on-the-fly is computationally demanding, in both simulation and real life. There can be frequent updates on the average speeds of the roads, which will also cause total trip times to change frequently. As known, link-state shortest-path algorithms are not scalable in the case of frequent updates in the link cost table due to the computational overhead of table reconstruction. We present the "RoutingAPI" to overcome this problem.

1) Checkpoints: In our congestion avoidance mechanism, instead of optimizing the routes globally in terms of trip times, we perform a local optimization. When a car calculates a shortest path to the destination in terms of distance at the beginning of its trip, we place checkpoints on the path with equal distance among them. In case any of the checkpoints turns out to be in a congested area, we push it further away on the path. In other words, checkpoints are dynamically changed along the way in order to solve this problem. Our algorithm calculates the least congested path only to the next checkpoint. When the car arrives there, it calculates a new least congested

path to the next checkpoint, and so on. This approach is more efficient and effective than global optimization for two reasons. First, cars can collect local congestion information much faster than global information since every car in the local area will be in the wireless communication range. Second, the local congestion information will be more up to date since by the time the car arrives at a distant road, congestion levels will change and a globally optimum route calculated at the beginning of the trip will end up being a suboptimal one later.

This approach also scales better than global optimization since cars will not have to store the congestion levels of all the roads in the map and local optimum route calculation will require less computational overhead. Additionally, by placing checkpoints on the shortest path in terms of distance, we minimize the number of checkpoints and prevent loops in the routes. Also this way, when our system becomes idle when there is no congestion or there are no cars around, vehicles will follow the shortest path in terms of distance. Therefore, our algorithm is highly tolerant to lack of or partial information.

2) RoutingAPI: Instead of finding all possible routes between two points during the simulation, we use an API to perform this operation at setup-time. The API stores all potential routes between two points, and returns the shortest *k* routes when asked. This connectivity data is generated offline; cars will just download it once and start querying the API (instead of generating it themselves). Note that the routes returned by the API are *k* best routes in terms of distance, calculated without the existence or assumption of any congestion information. Upon receiving these potential routes, a car will still need to use the congestion information available to it in order to select the least congested route.

The success of the API depends on two variables. First is the number of routes it returns, i.e., *k*. When *k* gets large, we ensure that more routes are considered by the vehicles when they reroute. As a result, a less congested route is less likely to be missed (i.e., left undiscovered). However, it also causes the size of the API database and the selection set of the cars to become larger. Hence, the computational cost required for rerouting will grow. The second parameter is the checkpoint interval, i.e., distance between the checkpoints as discussed in the previous section. A larger interval requires a larger *k*; otherwise most routes that make sense will be omitted from the selection. If the interval is too low, cars will be less flexible in their rerouting decisions. Existence of these parameters offers freedom with respect to the amount of resources available to the cars. For example, an unbounded *k* would mean a car will consider every possible route from source to destination, which might be desired assuming the car has enough computation power and space. On the other hand, it is best to set a reasonable *k* to run a giant simulation with limited resources. Similarly, checkpoint intervals can be set according to the reliable transmission range of the communication medium.

3) Least Congested Route Selection: We minimize the trip time between two checkpoints after getting *k*-best routes between them from RoutingAPI. Every time a car approaches the end of a road, it calculates what its roads of interest are in order to differentiate a less congested route from these *k* routes. The car then sends out a congestion request asking for these roads. Congestion responses received will be merged with the congestion information database of the car using Algorithm 1 as described before. The routing decision then becomes a set of arithmetic operations; the trip time for each of the *k* routes will be computed, and the car will choose the one with the least trip time as the chosen route. Let $RSet$ be the route database of the car that consists of the *k* candidate routes explained above. Each route R is a series of edges (roads) which are represented by E.

$$RSet = \{R_1, R_2, R_3, \ \dots \ , R_k\}, \quad R = \{E_1, E_2, E_3, \ \dots \ , E_i\}$$

Therefore, the selection of the least congested route from $RSet$ is performed using the following calculations:

$$R_{Chosen} = \min_{\forall R \in RSet} (\sum_{\forall E \in R} Length(E)/AvgSpeed(E))$$

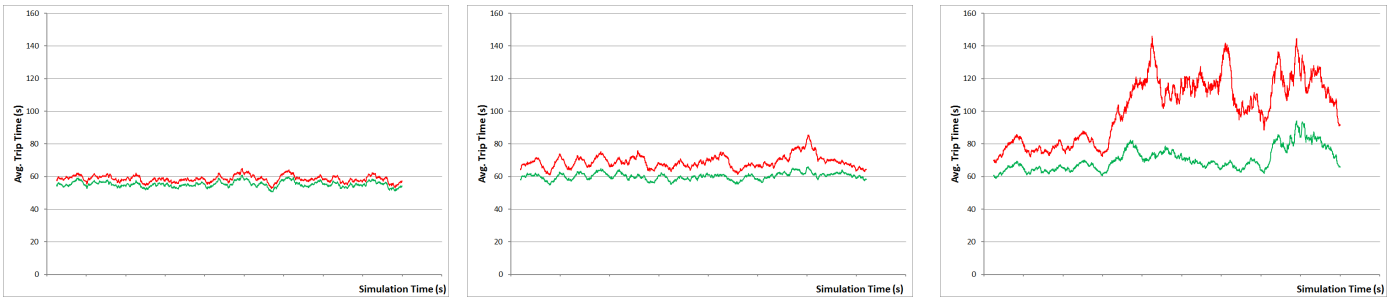
V. Evaluation

We used Veins [2] that couples SUMO [3] and OMNeT [4] simulators for our experiments. SUMO simulates vehicular traffic, whereas OMNeT takes care of communication between the vehicles. We simulated traffic in a Manhattan grid for 3600 simulation seconds (1 hr). We set random start and end points for each car entering the grid.

To better understand the effect of our congestion avoidance algorithm, we experimented with three levels of traffic: low (one car generated every 2 seconds), medium (one car generated every 1 second) and high (one car generated every 0.8 seconds). We observed no serious traffic congestion with low level of traffic. For medium amount of traffic, there was a slight increase in cars' trip times. There was a visible traffic jam when traffic volume was set to high, as average trip times increased by 100%. We can not report results for even higher volumes of traffic (e.g., cars generated every 0.5 seconds) because then traffic becomes so jammed that it is impossible to get reliable results within reasonable amount of time.

In Figure 2, we compare *offline* and *online* route assignment strategies. In the offline case, SUMO's user assignment module (duarouter) is used to calculate shortest-path routes, which are then fixed. Cars do not use V2V communications or re-plan their routes on the fly. In the online case, however, all cars use our VANET-based congestion avoidance mechanism. One can validate that there is an increase in average trip time as the traffic volume is increased; a rough calculation would show that the amount of time wasted in traffic goes from a mere 10-20% of vehicles' trip times to approximately 50% as the volume of traffic is increased. Although the benefits of using our system is marginal when the traffic volume is low (Figure 2(a)), reduction in trip times increases as the traffic volume gets bigger. The reason is that our system remains idle when there is no congestion to avoid to save vehicles from redundant calculations and communications. This way, as opposed to other systems in the literature, our system consumes vehicles' resources exactly as much as needed.

In addition to trip times, we also observed changes in the number/density of vehicles in the simulated area, as shown in Figure 3. Again, enabling or disabling our congestion avoidance system does not offer much when the level of traffic is low. Yet, it performs as well as (or slightly better than) being completely offline, thanks to the initial checkpoints. For high levels of traffic, we observe a significant reduction in the number of vehicles between 1500sec and 3000sec, since our system gets more active and aggressive as the congestion level increases.



(a) Traffic volume: low (b) Traffic volume: medium (c) Traffic volume: high

Fig. 2: Average trip times with respect to time. Red lines result when the congestion avoidance algorithm is not used, green lines depict average trip times when the algorithm is enabled. The axes share the same scale in all three graphs.

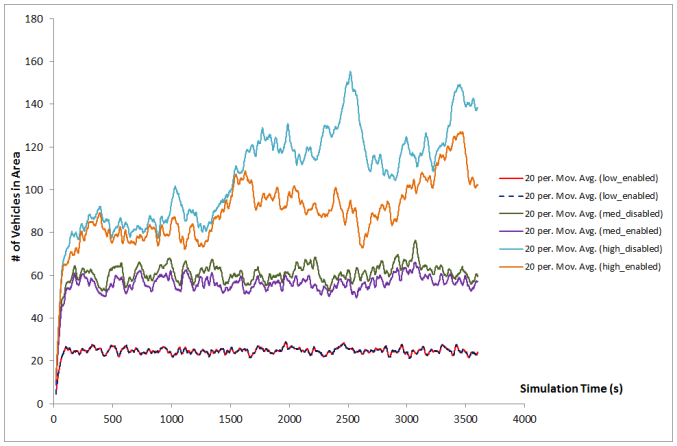


Fig. 3: Number of vehicles for different levels of traffic, with and without the congestion avoidance algorithm.

VI. Conclusion and Future Work

In this paper, we review existing VANET-assisted congestion avoidance mechanisms and propose a novel approach. We point out several design considerations and try to justify our choices through intuitive arguments and examples. We acknowledge that smart navigation systems with capabilities of avoiding traffic congestion are crucial in today's world and show that VANETs can be utilized to achieve this goal. The system we propose relies on sending information request messages whenever a vehicle desires or needs to learn more about upcoming roads and traffic. We believe that this scheme reduces the potential redundancies in frequent periodic broadcasting, and ensures a low computation overhead and scalable network load; this is crucial in a highly dynamic real-life traffic scenarios.

As part of future work, we plan to extend and evaluate this system by experimenting with changing request rates, changing percentage of cars that volunteer to use our system, and various BSM sampling frequencies/techniques. Also, we have made an implicit assumption in this paper that all vehicles honestly follow the protocols and algorithms to the best of their ability, e.g., they do not flood the system with excessive number of requests or supply bogus congestion information. We plan to investigate these issues in our future work as well.

References

[1] http://www.fastcompany.com/3022489/innovation-agents/self-driving-cars-let-go-of-the-wheel/.

[2] http://veins.car2x.org.

[3] http://sumo-sim.org.

[4] http://www.omnetpp.org.

[5] Dedicated short range communications (dsrc) message set dictionary. *WIP Standard J2735*, November 2009.

[6] R. Bauza, J. Gozalvez, and J. Sanchez-Soriano. Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In *IEEE 35th Conference on Local Computer Networks*, 2010.

[7] W. Chen, S. Zhu, and D. Li. Van: Vehicle-assisted shortest-time path navigation. In *IEEE MASS*, 2010.

[8] S. Fontanelli, E. Bini, and P. Santi. Dynamic route planning in vehicular networks based on future travel estimation. In *IEEE VNC*, 2010.

[9] P. J. He, K. F. Ssu, and Y. Y. Lin. Sharing trajectories of autonomous driving vehicles to achieve time-efficient path navigation. In *IEEE VNC*, 2013.

[10] J. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, July 2011.

[11] W. Kim and M. Gerla. Navopt: Navigator assisted vehicular route optimizer. In *5th International Conference on IMIS*, 2011.

[12] I. Leontiadis, G. Marfia, D. Mack, G. Pau, C. Mascolo, and M. Gerla. On the effectiveness of an opportunistic traffic management system for vehicular networks. *IEEE Transactions on ITS*, 12(4):1537–1548, 2011.

[13] M. Milojevic and V. Rakocevic. Distributed vehicular traffic congestion detection algorithm for urban environments. In *IEEE VNC*, 2013.

[14] M. Raya and J. P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[15] D. Schrank. *Urban Mobility Report (2004)*. DIANE Publishing, 2008.

[16] V. Verroios, K. Kollias, P. K. Chrysanthis, and A. Delis. Adaptive navigation of vehicles in congested road networks. In *ICPS*, 2008.

[17] Wireless LAN Working Group. Wireless access in vehicular environments. *IEEE Standards*, July 2010.

[18] L. Wischoff, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Sotis - a self organizing traffic information system. In *VTC*, 2003.

[19] Y. Yang and R. Bagrodia. Evaluation of vanet-based advanced intelligent transportation systems. In *ACM VANET*, 2009.

[20] J. Zhang, J. Xu, C. Zhu, W. Wang, and S. S. Liao. Constructing summarizations for v2v traffic data based on sampling methods. In *IEEE VNC*, 2010.