LOCUS: Efficient Urban Transportation using Localized Vehicular Coordination

Chandra Shekhar, Sudipta Saha
School of Electrical Sciences, Indian Institute of Technology Bhubaneswar, India
Email: {cs13, sudipta}@iitbbs.ac.in

***Abstract*—An efficient transportation system plays a significant role in the functioning of a smart city system. The ability of the vehicles to make the correct decisions regarding the selection of the right path at the right time to reach their destination is a crucial component of an efficient transportation system. However, dynamic decision-making during busy hours in congested cities becomes a challenging task due to frequent updates in the traffic status of the road fragments. While live coordination among the vehicles is a highly essential component, it suffers from serious scalability problems with the rapid rise in the number of vehicles as well as the area to be covered. Existing solutions to address this problem are mostly centralized and depend on efficient cloud services. Keeping in mind the serious vulnerabilities of the centralized solutions, in this work we design an efficient decentralized IoT-based framework LOCUS to enable the vehicle to coordinate with each other to solve the problem. LOCUS exploits recent advancements in the domain of *Synchronous-Transmission* (ST) to achieve efficient and fast local dissemination of traffic-status data enabling required re-planning of the path on-the-fly. Simulation-based evaluation shows that LOCUS achieves up to approx. 51% lesser average travel time compared to the existing best-known decentralized solution.**

***Index Terms*—VANET, Traffic-Management, Synchronous Communication**

I. Introduction

For efficient transportation in a large and busy city, appropriate planning of the paths of the vehicles is one of the most significant issues. Path-planning strategies have been studied in many works so far to optimize *travel time* or *travel distance* [14], [16]. The problem has been addressed well [2] for cases where the variation of traffic over the roads is negligible or quite low. However, dynamic changes in the traffic status, which is one of the most common phenomena, quickly make a pre-computed path almost useless. Moreover, the dynamic nature of traffic along with the possibility of unforeseen emergency situations at various places in a congested city also make it hard for the system to decide a new path from the source to the destination on-the-fly [9].

Existing works attempting to manage the situation under varying traffic significantly depend on cloud-based services, which are fundamentally centralized in nature [15]. Unfortunately, a centralized solution always suffers from a set of inherent problems all rooted in the well-known issue of *single-point-of-failure*. In particular, assistance from cloud services may get drastically disrupted due to various reasons, such as possible targeted attacks, or high congestion in the network, etc. In addition, during disastrous scenarios where dynamic path planning by the vehicles happens to be a crucial requirement, cloud-based services may not even be available due to possible disruption of the necessary infrastructure. Considering such adverse situations, there have been attempts in various works to design decentralized, self-sufficient edge-based path-planning strategies that can operate without support from any specific heavy-weight infrastructure, e.g., PANDORA [1], DIVERT [11], and RIDER [9].

Existing works in this direction [1], [16], [14] have shown that the updated congestion information, although a highly necessary component, needs to be made accessible to the vehicle in a planned and systematic way. It is shown that, while too little information is inadequate to obtain the desired outcome, too many updates, e.g., at every junction in the road network [10], may even mislead the vehicles unless the updates are appropriately linked and structured. The effect of making current traffic-congestion data available to vehicles during their journey in an uncoordinated or unlinked fashion leads to the possibility of misleading the vehicles and provoking them to select the wrong or inefficient path. In a nutshell, to manage the dynamic status, it is essential for the vehicles to obtain sufficient data regarding the traffic status of the surrounding roads in real-time. In addition, the data must be inter-linked with each other instead of being isolated. However, achieving such linked or coordinated data in real-time as well as in sufficient amounts is a challenging task in a traditional decentralized setting.

Objectives: In this work, we aim to accomplish two distinct objectives: (a) Design and develop *a decentralized setup* for intelligent transportation that is capable of providing periodic and coordinated information to the vehicles in real-time. (b)Ensure scalable behavior amidst a large number of vehicles. In fact, since the number of vehicles can be quite unbounded, we ensure that the proposed framework is independent of it.

Communication among the participating entities in a decentralized system is the primary component of any sort of coordination. However, the existing CSMA/CA-based strategies, while executed in a large-scale decentralized system consisting of many nodes willing to share their data, suffer from significant delays in communication because of the considerable chance of collisions among the packets. Moreover, the use of the traditional back-off-based strategies in CSMA/CA for resolving collisions makes these strategies almost incapable of real-time communication, especially when the data traffic is high, which is quite expected under vehicular conges-

979-8-3503-8741-4/24/$31.00 ©2024 IEEE

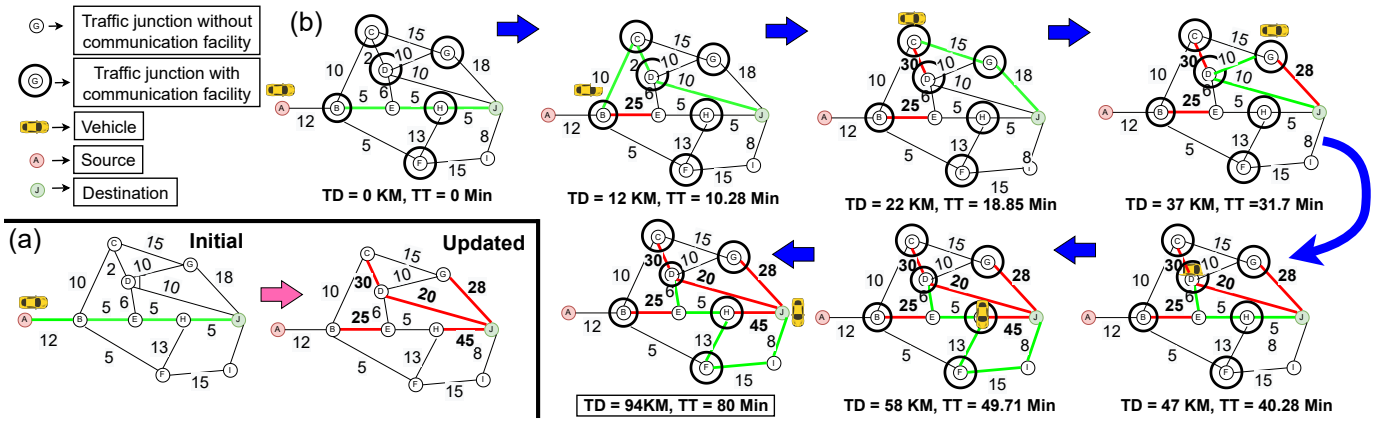

Fig. 1. Illustration portraying the implication of making congestion information available to the vehicle.

tion. *Synchronous transmission* (ST)-based communication mechanisms, on the contrary, have been quite successful in accomplishing real-time communication and data-sharing as demonstrated by several recent works [5], [12], [3], [7]. These strategies avoid CSMA and rely on precise time-bounded operations. In this work, we propose a *LOcalized vehicular Coordination-based strategy to assist Urban Transportation System* (LOCUS). It makes efficient use of ST for coordination among the vehicles to ensure real-time availability of the necessary congestion status and dynamic re-computation of the path to the destination on-the-fly.

The main contributions of the paper are as follows.

- We propose a lightweight, self-reliant IoT-assisted decentralized framework LOCUS which enables the vehicles to obtain traffic-congestion-related information in real-time and dynamically adjust the path to their destinations.
- We demonstrate how ST-based data-sharing can be efficiently employed to enable the vehicles to share their journey experiences with one another on-the-fly considering their large number as well as their span over a large area.
- We implement the proposed strategy in Contiki OS and evaluate its performance in the simulator Cooja.

The rest of the paper is organized as follows. Sec. II provides a brief background, and Sec. III provides a detailed description of the design of the proposed IoT-based framework and the coordination strategy. Sec. IV reports a detailed evaluation of LOCUS w.r.t. the existing best-known decentralized strategies for dynamic path-planning.

II. Background

We extensively make use of ST-based communication strategies. In the following, we describe two ST-based data-sharing strategies that we employ in the design of the coordination strategy used in LOCUS.

Glossy: Glossy [8] exploits ST for quick sharing of data from a designated node (initiator) to all the other nodes in a wireless multi-hop setting. The initiator node starts the process by broadcasting the data-packet. The neighboring nodes, subsequently, forward the packet using ST which triggers special physical layer phenomena called *Capture-Effect* (CE) in the receiving nodes. Due to CE, the packets having stronger signal wins even though multiple packets arrive together at the receiver. Glossy serves the purpose of both one-to-all dissemination as well as system-wide time-synchronization.

ST has been also exploited for realizing complex data-sharing patterns [12], [7], [3], [6] in IoT systems. These works have been used to support data-sharing in dynamic networks too, e.g., VANET [13], Multi-Drone Systems [4]. Most of these works exploit packet-level TDMA in an ST-based framework to accomplish all-to-all data-sharing. However, all these works use a global TDMA schedule where each node is mapped to a unique time slot. It makes these strategies unsuitable to support real-time data-sharing in large deployments, e.g., data-sharing in vehicular networks which naturally spread over a substantial part of a city and are comprised of many nodes. In the current work, we employ a recently invented strategy LiteCast [5] to overcome the problem.

LiteCast: LiteCast, to solve the aforementioned issue, employs a very short-sized schedule which is of the order of the average degree of the nodes in the targeted network. The reduction in the number of slots causes conflicts and collisions among the packets, which are resolved through clever use of CE as well as various heuristics. However, to balance the requirements, LiteCast supports efficient many-to-many dissemination of data only among the neighboring nodes in the system. In the following, we provide the complete design of the strategy.

III. Design

Fig. 1 illustrates a possible scenario with a road network having nine junction points. It is assumed that the traffic congestion data is made available to the vehicles at almost all the junctions. However, the shared data is purely local and is independent of what is available at the next or other junctions, which leads to certain problems in path planning. Fig. 1(a) captures all the updates in the road-network together. A vehicle starts from junction A (*source*) and targets to reach junction J (*destination*) minimizing travel distance and time.

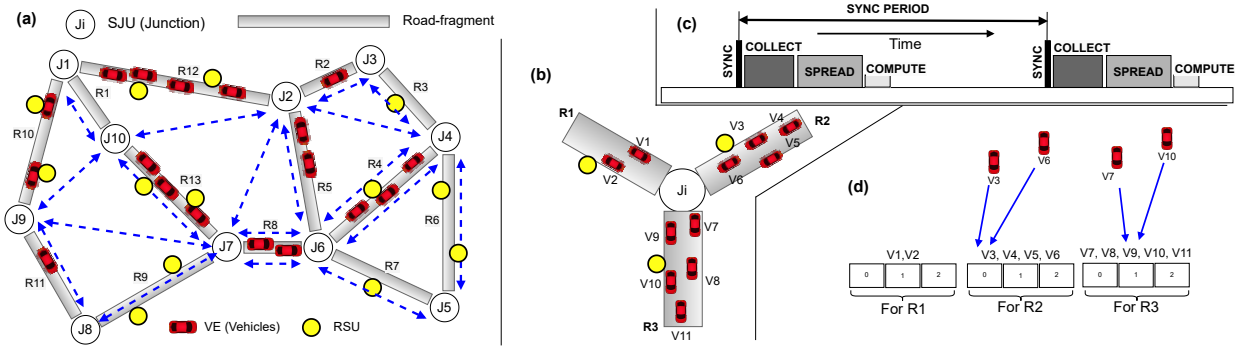

Fig. 2. The IoT-setting used in LOCUS. (a) A scenario with SJUs, VEs, and RSUs. (b) An SJU with three road fragments and some VEs. (c) Timing sequence of the components. (d) Hashing-based recording of the feedback from the vehicles by the SJU shown in (b).

If it cannot see all updates together or is not even able to see a sufficient part of them collectively, it fails to make the correct decision to accomplish its goal. Applying Dijkstra's algorithm to the graph shown in the figure with the updated weights, the best possible travel distance and time can be computed as 40 km and 34 min, respectively. However, Fig. 1(b) shows the possible consequence of providing unlinked data to the vehicle. It leads to the total travel distance and time of 94 km and 80 min, respectively, which are almost 2.5 times the best achievable values.

Algorithm 1 ALGORITHM: VEHICULAR COORDINATION

Every VE initiates G with the map of the area.
Every VE gets the shortest path using Dijkstra's algorithm.
Following executes periodically -

(SYNC)

Synchronization using ST-based flooding

(COLLECT)

Every SJU initializes a list D
for Every SJU **do**
    Send Probe to the VEs.
    List of road-fragments and m is supplied.
    A VE calculates slot number $i = m \times (j-1) + v\%m$
    SJU collects feedback by all the VEs
    for Every R_j **do**
        Isolate the contributions from the VEs
        $d_j \leftarrow$ Average of all contributions
        Append d_j in D
    end for
end for

(SPREAD)

Initialize local data-store L
for Every SJU j **do**
    Set the data to be shared as D_j
    Set slot size t based on average road-connectivity and expected degree of local-coverage.
    Execute LiteCast using D_j and t
end for
Every SJU store obtained lists from nearby SJUs in L

(COMPUTE)

for Every VE **do**
    Use the data obtained in L to update G
    If significant updates, use Dijkstra's algorithm to recompute the shortest path.
end for

The above example makes it clear that for successful on-the-fly planning of the path, the vehicles need to have a complete picture instead of only knowledge about the status of their current location. However, gathering the data from the whole city in real-time is also not quite possible in a decentralized setting. The design of LOCUS is done keeping these conflicting requirements in mind.

Fig. 2 shows an overall setting used in LOCUS. We assume three types of entities: *Static Junction Unit* (SJU), *Road Side Unit* (RSU), and *Vehicles* (VE). Each of these units is considered to be equipped with an IoT device that is capable of communicating with each other using a suitable technology like 802.15.4 or LoRa. RSU is fundamentally similar to SJU with simpler configuration and resources as they have lesser responsibilities. In particular, an SJU acts as the source of the data while an RSU only forwards the data they receive so that the network remains connected. An SJU is installed only in a road junction, where the vehicles get an opportunity to change their decision regarding the path they have already computed. For large road fragments, we depend on the RSUs for connectivity.

The process works through periodic invocation of the following three steps (a) *COLLECT:* The SJUs collect the necessary data from the VEs and compute the necessary information to be shared next with the VEs, (b) *SPREAD:* The SJUs with the help of the RSUs spread the information among each other. (c) *COMPUTE*: The VEs recompute their path to their destinations and decide to take a different road fragment at the current junction if needed. To enable ST-based operation, LOCUS uses an instance of Glossy in the being of every iteration (SYNC). The timing diagram in Fig. 2(d) depicts the timing sequence of all these steps. Algorithm 1 shows the complete picture with all the components together.

The VEs are considered to have a physical map of the complete area to be covered. Each road-fragment, and junction point (SJU) bears a unique identity which is included in the map. A sample scenario is shown in Fig. 1(a). The VEs also have their unique identities. However, since the number of vehicles can be quite unbounded we do not directly use it in LOCUS. In the COLLECT phase, when a VE comes near an SJU, it receives a probe message where the SJU asks all the nearby VEs to share their information about the road-fragments they recently traversed. In particular, an SJU asks about only those road fragments with which it is directly connected. However, keeping in mind the huge number of the VEs that would be willing to provide their feedback, a number

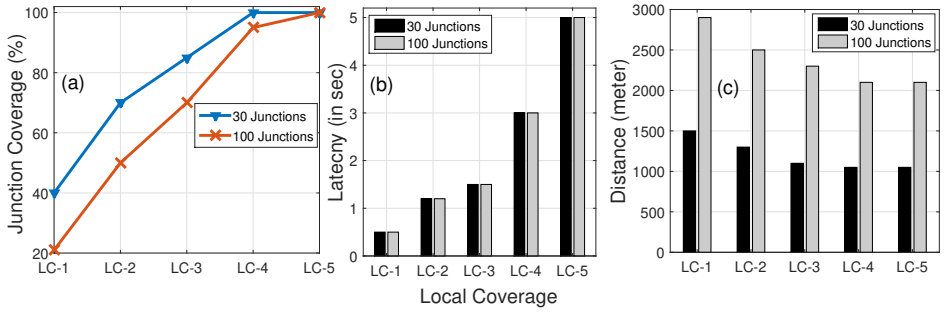

Fig. 3. (a) Effect of different degrees of local coverage obtained through LiteCast. (b) Latency to complete different degrees of local coverage. (c) Travel distance under different degrees of local coverage among the SJU.

of slots are offered for each of the road-fragments.

In particular, the total number of time slots t_s offered by an SJU s is set as a fixed multiple m of the number of road-fragments r connected with s (i.e., $t_s = m \times r$). All the VEs willing to provide their input for a specific road-fragment at s get m slots. The probe packet provides all this information. A VE uses a simple modulo-based hashing strategy to decide in which slot it should provide its input. In particular, VE having id v provides its feedback for road-fragment R_i in the slot number $i = m \times (j-1) + v\%m$ where R_i is the global identification number of the road-fragment and it is the j-th road-fragment in the list of the road-fragments connected with SJU s. Due to the scarcity of time slots compared to the number of VEs, a conflict is inevitable among multiple VEs attempting to provide their feedback in the same slot. Here, we rely on the use of ST. Specifically, under CE as an artifact of ST-based communication, most of the time an SJU receives a sufficient number of feedbacks from multiple VEs.

The algorithm for computing the feedback can be quite sophisticated. However, to gain proof of the concept, in this work, we use a simple strategy. It is calculated in the form of a value in the range of 0 to 100. A high value means high congestion, indicating a higher time requirement to move through the road-fragment. A low value indicates that the vehicle experienced a smooth journey through the road-fragment. After acquiring multiple feedbacks from different VEs the SJU averages them and makes a final value ready for each of the road-fragments it is connected with. In the SPREAD phase, the SJUs and RSU run an instance of LiteCast with a certain target degree of local coverage determined by the parameter *Local-Coverage* (LC). For LC=i, each of the i-hop neighbors of each of the SJU receives data from one another with reliability above 99.9% through an instance of LiteCast [5]. Note that in the SPREAD phase, the VEs also participate just as a silent receiver and forwarders of information. Finally, in the COMPUTE phase, the VEs use their local map data along with the local updates to recompute the current best path using Dijkstra's shortest path algorithm.

In the whole design, it's quite interesting to study the effect of the parameters m, and LC. To keep it simple and small, we keep the value of m to be 3 in our simulation setup. The effect of LC is quite important, as it directly relates to the correctness of the re-computed path as well as the time delay necessary to make the information available to the VEs. In the next section, we study these issues.

IV. Evaluation

We first demonstrate the effectiveness and advantage of the proposed concept of path planning based on localized data. Next, we demonstrate the superiority of LOCUS in comparison to the existing best-known strategies for decentralized path-planning. To fulfill the goal, we implement the strategy in Contiki OS for TelosB motes and simulate it in the default simulator of Contiki, i.e., Cooja. IEEE 802.15.4-based communication technology is used for all simulations. Each of the junctions is equipped with one SJU. A set of VEs start their journey from one corner point of the experiment area and target to reach the corner point on the other side. Cooja in general does not support the mobility of the nodes. Hence, we tweak the Java-based front-end of Cooja to support mobility for simulation of the behavior of the vehicles. Each experiment is repeated for at least 1000 times. The metrics are computed in each of the VEs and SJUs. Finally, the results are prepared based on the average of the metrics over all the components and all iterations. SJUs, RSUs, and VEs are all implemented as sky motes in Cooja.

A. Local-coordination

How much local coverage is required to accomplish a reasonably good performance by LOCUS is studied first. Fig. 3(a) first shows the number of neighboring SJUs LiteCast covers when executed under different values of LC. LC-i implies i-hop local-coverage around every SJU. The performance of LOCUS is next studied in two different settings: one with 30 junctions and 200 vehicles over an area of 2500 X 2500 square meters, and the other with 100 junctions and 300 vehicles over an area of 5000 X 5000 square meters. Fig. 3(c) shows the results. Note that the density of the vehicles in both settings is almost the same. Therefore, under both settings, LiteCast takes almost a similar time to achieve LC-1 to LC-5 as shown in Fig. 3(b).

B. Comparison with State-of-the-art

We compare the performance of LOCUS with the existing state-of-the-art decentralized path-planning strategies PANDORA [1], and RIDER [9]. In all these experiments we set

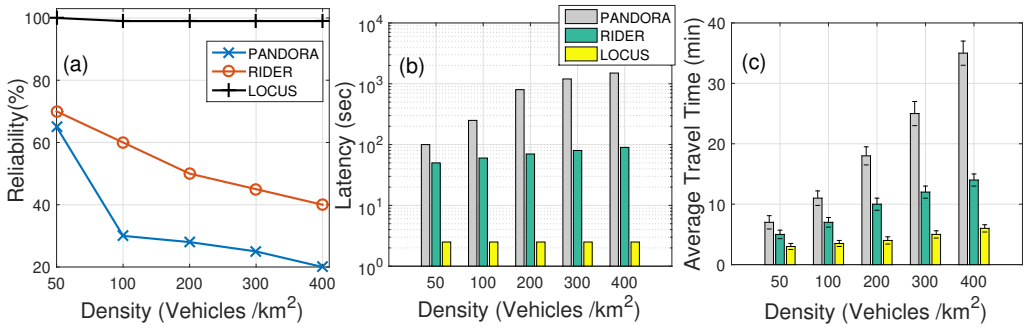

Fig. 4. Performance comparison of LOCUS with existing strategies. Comparison of reliability in obtaining vehicular data (a), time to acquire the data (b), and average travel time of the vehicles (c).

the area of an experiment at 3000x3000, with 40 SJUs and varying numbers of VEs. The sharing of traffic data by VEs is one of the prime sources of information in all three works. So, we first check how reliably the data is conveyed from a vehicle to all the other vehicles within a radius of 500 meters under varying vehicular density. The metric reliability is calculated as the ratio (in percentage) of the number of the road-fragments whose data have been acquired by a VE through the strategy w.r.t the total number of road fragments present within a radius of 500 meters. The results in Fig. 4(a) show that LOCUS quite reliably obtains the data even when the density of the vehicles is quite high. However, PANDORA and RIDER show a drastic fall in reliability which is one of the prime causes of their inability to reduce travel time compared to LOCUS as shown in Fig. 4(a) especially when the density of the VEs rises.

The efficiency of the data dissemination among the VEs with the help of the SJUs is also reflected through the latency values plotted in Fig. 4(b). It shows the time needed by a VE to obtain up-to-date data regarding the road-fragments within a radius of 500 meters. It can be seen that the use of ST-based communication as well as the appropriate use of static SJUs enables LOCUS to obtain the data much faster compared to PANDORA and RIDER which is another reason for the superiority of LOCUS in reducing travel time as well as scalable operation. In summary, LOCUS achieves up to 51% and 79% lesser travel-time compared to PANDORA and RIDER.

V. Conclusion

In this work, we propose an IoT-based decentralized framework, LOCUS to support local coordination among the vehicles to enable dynamic computation of the path to their destination. In particular, the proposed strategy enables the vehicles to efficiently share each other's travel experiences, which are further used by them for the re-computation of the shortest path on-the-fly. LOCUS achieves scalability by introducing an appropriate hybrid mechanism where the vehicles interact with a static setting. Furthermore, we also show that only local-coordination through local-sharing of data is quite good enough to achieve the desired goal. We demonstrate that LOCUS successfully achieves approx. 51% lesser travel time compared to the existing best-known decentralized solution.

References

[1] A. M. de Souza, N. L. S. da Fonseca, and L. Villas. A fully-distributed advanced traffic management system based on opportunistic content sharing. In *Proceedings of IEEE ICC*, 2017.

[2] A. M. de Souza, R. Yokoyama, A. Boukerche, G. Maia, E. Cerqueira, A. A. Loureiro, and L. A. Villas. Icarus. *Comput. Netw.*, 110(C):118–132, dec 2016.

[3] J. Debadarshini, M. Kausik, and S. Saha. Structure-adaptive many-to-many data-sharing for internet-of-things. *IEEE TNSM*, pages 1–13, 2024.

[4] J. Debadarshini and S. Saha. An iot-assisted efficient framework for multi-drone conveyance system. In *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pages 6426–6431, 2023.

[5] J. Debadarshini and S. Saha. Litecast: Flexible scalable and uniform local data-sharing in real-time. In *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–10, 2023.

[6] J. Debadarshini and S. Saha. Synccast: Real-time self-adaptive and fault-tolerant all-to-all data-sharing in iot. In *2023 IEEE 31st International Conference on Network Protocols (ICNP)*, pages 1–11, 2023.

[7] J. Debadarshini, M. Tummala, S. Saha, O. Landsiedel, and M. C. Chan. Timecast: Real-time many-to-many data-sharing in low-power wireless distributed systems. *IEEE Systems Journal*, 17(4):5726–5737, 2023.

[8] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *Proceedings of ACM/IEEE IPSN*, 2011.

[9] T. S. Gomides, R. E. De Grande, F. S. Souza, and D. L. Guidoni. Rider: Proactive and reactive approach for urban traffic management in vehicular networks. In *Proceedings of 16th IEEE DCOSS*, 2020.

[10] M.-A. Lebre, F. Le Mouel, and E. Menard. Partial and local knowledge for global efficiency of urban vehicular traffic. In *Proceedings of IEEE 82nd VTC2015-Fall*, 2015.

[11] J. Pan, I. S. Popa, and C. Borcea. Divert: A distributed vehicular traffic re-routing system for congestion avoidance. *IEEE TMC*, 16(1), 2017.

[12] S. Saha, O. Landsiedel, and M. C. Chan. Efficient many-to-many data sharing using synchronous transmission and tdma. In *Proceedings of 13th IEEE DCOSS*, 2017.

[13] C. Shekhar, J. Debadarshini, P. K. Singh, and S. Saha. A lightweight iot-based framework for vehicular ad hoc network (vanet). In *2023 15th International Conference on COMmunication Systems NETworkS (COMSNETS)*, pages 19–24, 2023.

[14] M. Wang, H. Shan, R. Lu, R. Zhang, X. Shen, and F. Bai. Real-time path planning based on hybrid-vanet-enhanced transportation system. *IEEE Transactions on Vehicular Technology*, 64(5):1664–1678, 2015.

[15] S. Wang, S. Djahel, and J. McManis. An adaptive and vanets-based next road re-routing system for unexpected urban traffic congestion avoidance. In *Proceedings of IEEE VNC*, 2015.

[16] H. Wu, H. Zhou, J. Zhao, Y. Xu, B. Qian, and X. Shen. Deep learning enabled fine-grained path planning for connected vehicular networks. *IEEE Transactions on Vehicular Technology*, 71(10):10303–10315, 2022.