Adaptive Broadcast Scheduling Scheme for High-definition Map Tile Dissemination in Vehicular Networks

Madhuri Annavazzala, Surpiya Dilip Tambe, Antony Franklin A., and Bheemarjuna Reddy Tamma
Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad
Email: {cs21mtech02003, cs18resch11002, antony.franklin, tbr}@iith.ac.in

***Abstract*—Autonomous vehicles require precise and reliable data for critical functions like localization and navigation. High-Definition (HD) maps are essential, segmented into various layers (each with varying roles and lifespans) and vertically structured into 'tiles' that encapsulate these layers. Efficiently disseminating these tiles via Road Side Units (RSUs) in Vehicle-to-Everything (V2X) networks is challenging. An efficient dissemination method must minimize both the Turn Around Time (TAT)—the time from a tile request to receipt—and the incidence of schedule misses, which is when a vehicle does not receive a tile before its deadline, impacting reliability and performance. This paper introduces an Adaptive Broadcast Scheduling (ABS) scheme tailored for HD maps, factoring in layer priority, request popularity, and deadlines. Experimental studies show that ABS outperforms conventional periodic broadcast by cutting schedule misses by over 50% and reducing TAT to milliseconds.**

***Index Terms*—Adaptive Scheduling, Broadcast Schemes, High-definition Map, Optimization, Vehicle-to-Everything (V2X).**

I. Introduction

AUTONOMOUS vehicle navigation relies on High-definition (HD) maps, which provide intricate, centimeter-level environmental details essential for safe operation [1]. However, the development and utilization of HD maps present significant challenges due to their extensive data requirements, impacting both uplink and downlink of mobile networks. Conceived at a 2010 Mercedes-Benz workshop [2], these maps transcend conventional navigation tools by providing real-time, granular insights into road layouts, markings, signs, and conditions, enabling autonomous systems to navigate complex traffic with efficiency and make split-second decisions to ensure safety. In 2012, the introduction of Local Dynamic Maps (LDM) further fueled the discussion and development of HD maps by outlining four layers that stored both static and dynamic information locally [3]. Although agencies have been working on creating HD maps such as Navigational Data Standard (NDS) [4] and OpenDRIVE [5] which usually have Point Cloud Data (PCD)[1] and Open Street Map (OSM)[2] formats, there is no standardized framework dictating the specific number of layers requisite within an HD map yet.

Fig. 1 shows that HD maps, unlike traditional 2D navigation maps, consist of multiple layers [6]. A vertical section of this 3D grid is a 'tile', while a horizontal section is a 'layer'. Each layer has distinct characteristics in terms of data, scale, and temporal relevance. The bottom layer, updated monthly/weekly/hourly, includes static information like roads and buildings is the most important layer without which even basic navigation would not be possible. In contrast, the top layer, updated in real-time, contains dynamic data such as pedestrian and vehicle movements. Update frequencies for intermediate layers range from minutes to weeks, depending on their role. For autonomous vehicles, timely access to these layers is critical for effective decision-making, necessitating a scheduling system in wireless environments that accounts for vehicle mobility, channel usage, deadlines, and layer importance.

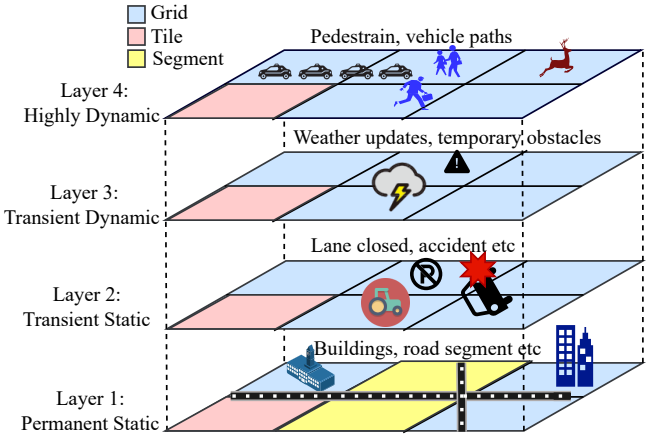


Fig. 1: An example of HD map with four layers.

In 5G New Radio Vehicle-to-Everything (NR-V2X), Road Side Units (RSUs) primarily use unicast/multicast or broadcast mechanisms for wireless information dissemination [7]. Unicast and multicast mechanisms, which operate on request-reply and publish-subscribe patterns respectively, consume significant bandwidth due to signaling overhead and can lead to redundancy when multiple vehicles request the same data. In contrast, broadcast mechanisms, transmitting information once for access by all vehicles, are more scalable in dense NR-V2X environments as they do not require resource allocation for each vehicle.

HD maps require extensive data, impacting both uplink and

[1]https://pointclouds.org/documentation/tutorials/pcd_file_format.html
[2]https://www.openstreetmap.org/

downlink of mobile networks. Unicast and multicast are less suitable for densely populated areas and are generally used for private data, while broadcast mechanisms are bandwidth-efficient for public data like HD maps. However, traditional broadcast mechanisms have limitations, such as delay in receiving requested data (schedule misses) and unnecessary data dissemination. To address these issues effectively, we propose the Adaptive Broadcast Scheduling (ABS) scheme. This scheme not only assesses the size and temporal aspects of each tile but also actively evaluates the demand for different tiles. By incorporating the popularity of tiles into its scheduling algorithm, ABS dynamically schedules tiles in the broadcast channel in real-time, ensuring a more efficient and demand-responsive dissemination of data.

The following are the key contributions made in this work:

- We devise a mathematical model that takes into consideration the bandwidth of RSU, speed of the vehicles and the HD Map size to encapsulate a HD map dissemination system. This model further evolves into an optimization framework, highlighting the NP-hard nature of the associated scheduling algorithm.
- We also propose a heuristic ABS scheme that considers the priority of each layer, validity constraints (deadlines), and popularity of the requested tiles to provide the dissemination of tiles by increasing the hit rate in dense vehicular environments.

The rest of the paper is organized as follows: Section II provides the related work. Section III presents the system model. Section IV presents the proposed Adaptive broadcast tile scheduling (ABS) algorithm in detail along with the devised optimization model. Section V contains the simulation setup used for performance evaluation. Section VI presents the simulation results. Finally, Section VII concludes the paper with some future research directions.

II. Related Work

The existing literature primarily focuses on how vehicles request HD map data from RSUs and the strategies for caching this data for quick delivery. Studies like [8, 9, 10, 11], emphasize pre-fetching and storing HD map content in cache using methods like Long short-term memory (LTSM), Area of Interest (AoI), data scope, cooperative caching techniques, and distributed Multi-Agent Multi-Armed Bandit learning (MAMAB) methods respectively.

These approaches mainly utilize unicast patterns, which may not be efficient for real-life scenarios, especially considering the significant data volume inherent in real HD maps, such as Google's HD map, which holds around 1 GB of data for every mile [12]. This considerable data size presents unique challenges in terms of efficient storage, retrieval, and wireless dissemination, underscoring the need for more specialized and robust data handling techniques in this context. Even in proposed scheduling approaches in [13], despite the focus on time-critical data dissemination in some literature, there's a notable gap in discussing specific scheduling techniques for HD map data, which also includes essential aspects like the size and priority of different layers. This highlights the need for exploring wireless dissemination strategies for HD map data. The closest research addressing the use of broadcasting for map tile dissemination, with the advent of digital terrestrial broadcasting in Japan [14]. However, while utilizing broadcasting for map tiles, it veered towards proposing a new cache system rather than a scheduling algorithm. It introduces "scope" and "mobility specification" for effective pre-fetching and replacement of broadcast data. Although it recognizes the limitations of basic broadcasting, it stops short of suggesting a new scheduling algorithm to overcome the challenges posed by periodic broadcasts. Furthermore, the focus was on 2D navigational maps like Google Maps, not on broadcasting HD maps. This underscores the necessity for developing scheduling algorithms that specifically account for the distinct characteristics of HD maps. To the best of our knowledge, this work is pioneering in addressing this particular need.

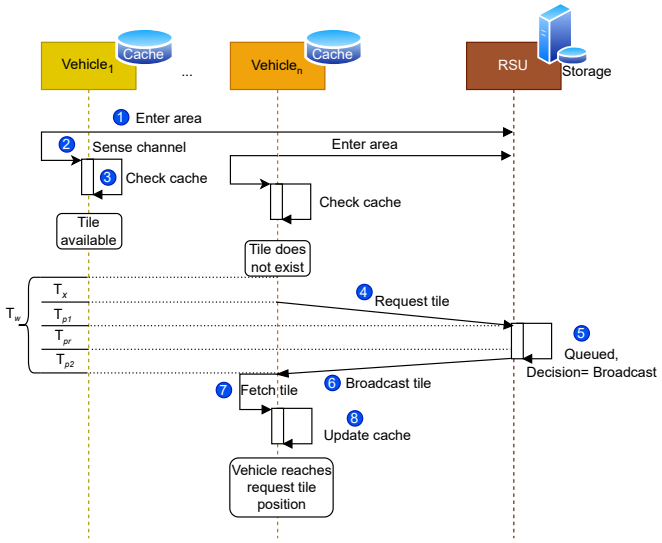


Fig. 2: Working of the HD Map dissemination system.

III. System Model

This section outlines the system architecture for HD map dissemination, as depicted in Fig. 2. We assume that the RSU is equipped with NR-V2X radio working in Mode 1, and serves as the HD map dissemination server. It stores the map divided into tiles of different sizes and two layers (l_{osm},l_{pcd}) each with different size and validity within its local storage. We assume that, up to 80% of the RSU's bandwidth, denoted as B_R is dedicated to disseminate the layers of HD Map over the broadcast channel using the PC5 interface, facilitating Vehicle-to-Interface (V2I) communication. The rest is allocated for other communications, including handling unicast requests for tiles from the vehicles.

The set of moving vehicles denoted by $\mathcal{V} = \{v_1, v_2, \cdots, v_v\}$. When a moving vehicle v, requires one or more layers $l \in \mathcal{L}$ for an upcoming road segment, it initially searches its local cache using the location of the upcoming path. If the layer/tile is either absent or outdated, the vehicle monitors the broadcast channel

TABLE I: Key Notations

Notation	Description
$\mathcal{V}$	Set of v vehicles.
$\mathcal{S}$	Set of s slots.
$\mathcal{R}$	Set of r tile requests.
N	Number of requests in $\mathcal{R}$
$\mathcal{L}$	Set of l layers.
Q	Maximum priority queue to order tile requests.
O	Ordered set of tile requests.
T_x	Transmission delay of vehicle.
T_{p1}	Propagation delay from vehicle to RSU.
T_{p2}	Propagation delay from RSU to vehicle.
T_{pr}	Queuing and processing delay at RSU.
T_{ch}	Time to acquire channel.
T_{tat}	Turn Around Time (TAT) of the requested tile.
T_{pos}	Time required by the vehicle v to reach the requested tile position.
T_b^r	Time at which a tile request r appears on the broadcast channel.
T	Complete duration.
t	Current time.
t_r^a	Arrival time of the tile request r.
t_r^d	Deadline of the tile request r.
T_P	Broadcast period.
B_R	Physical data rate of RSU.
B_v	Physical data rate of vehicle.
D_{RV}	Distance between RSU and vehicle.
D_{VL}	Distance between vehicle and requested tile.
$speed_v$	Speed of the vehicle V (in km/hr).
$size_r$	Size of tile request packet (in bytes).
c	Speed of light.
$Popularity(r)$	Number of times vehicles requested r.
$Priority(r)$	Relative importance of the layer.

to check if the needed data is currently being broadcasted by the RSU. In the event the data is not being broadcasted, the vehicle then sends a unicast request to the RSU. The set of all vehicle requests denoted by $\mathcal{R} = \{r_1, r_2, \cdots, r_r\}$. These requests are generated for a duration T. For simplicity, this duration T is divided into set of time slots denoted by $\mathcal{S} = \{s_1, s_2, \cdots, s_s\}$. This request $r \in \mathcal{R}$ includes its own location, the specific tile's location and its speed $speed_v$, communicated at a physical rate of B_v. This allows the vehicle to engage in pre-fetching the tile by transmitting its forward latitude and longitude coordinates to the RSU, which is then converted into tile IDs. The assignment of these tile IDs is an internal process at the RSU and is not disclosed to the vehicle. The vehicle takes T_x time to acquire the channel by following dynamic scheduling scheme [7] of Mode 1 and transmits the request. It is given by $T_x = T_{ch} + \frac{size_r}{B_v}$. RSU receives this request r after a propagation delay of T_{p1}, where $T_{p1} = \frac{D_{RV}}{c}$.

The RSU processes each received tile request through a structured approach consisting of three distinct phases: decision, queuing, and scheduling. It takes T_{pr} time to process the tile request and the broadcast reply reaches the vehicle after a delay of T_{p2}. The entire duration, encompassing the time from when a vehicle requests a tile, to the moment it successfully receives it from the RSU, is collectively referred to as the TAT, T_{tat}, which is given by $T_{tat} = T_x + T_{p1} + T_{pr} + T_{p2}$.

For every request r, T_{tat} must be strictly shorter than the time it takes for the vehicle v to reach the position of the requested tile, T_{pos}. Which means that the RSU has to schedule every request r, at some slot s in the broadcast channel, at time t such that slot s occurs before the vehicle reaches tile position. $T_{pos} = \frac{D_{VL}}{speed_v}$. If this condition is not met, the situation is classified as a schedule miss. This requirement ensures that the information received is timely and relevant for the vehicle's navigation. $T_{tat} < T_{pos}$.

IV. Proposed Work

In this section, we develop an optimization model for HD Map scheduling and prove its NP-Hardness. The optimal solution requires prior knowledge of the request order, which is typically unfeasible in real-world scenarios. Therefore, given these constraints and the complexity of the problem, we propose a heuristic scheme (ABS), as a more practical and implementable approach and present it in detail.

A. Optimization Model

The objective is to minimize the sum of unscheduled requests across all requests $\in \mathcal{R}$, taking into account the arrival and deadline times for each request. The decision variable is $X_{r,s}^t = 1$, if request r is scheduled in slot s at time t, otherwise 0.

$$\min \sum_{r \in \mathcal{R}} \left(1 - \sum_{t=t_r^a}^{t_r^d} \sum_{s \in \mathcal{S}} X_{r,s}^t \right) \tag{1}$$

subject to the following constraints:

- Slot occupancy constraint: Each slot s can hold only one tile at a time t. This constraint ensures that two requests cannot be scheduled in the same slot s at the same time t.

$$\forall s \in \mathcal{S}, \forall t : \quad \sum_{r \in \mathcal{R}} X_{r,s}^t \leq 1. \tag{2}$$

- Request scheduling constraint: Every request needs to be scheduled at least once between its arrival time and deadline.

$$\forall r \in \mathcal{R} : \quad \sum_{t=t_r^a}^{t_r^d} \sum_{s \in \mathcal{S}} X_{r,s}^t \geq 1. \tag{3}$$

B. NP-Hardness of the Problem

A known NP-Hard problem, a satisfiability (SAT) problem is a decision problem that asks whether there exists an assignment of boolean values to variables that satisfies a given boolean formula [15]. Our problem can be broadly categorized as a multi-objective scheduling problem, where the approach not only tries to maximize the schedule hits, but also ensure that it maintains the priority while doing so. The constraints can be re-written into those of a boolean formulation, where we need to prove its feasibility.

- For the slot occupancy constraint, for each time t and each slot s, we introduce clauses that essentially state if one request r is scheduled in slot s at time t, then no other request $r' \neq r$ can be scheduled in slot s at the same time. The clause for two requests r_1 and r_2 would be:

$$\neg X_{r_1,s}^t \vee \neg X_{r_2,s}^t. \tag{4}$$

- For the request scheduling constraint, we need to ensure that for each request r, there is at least one t and s such that $X^t_{r,s}$ is true. This is represented by a clause for each request that is a disjunction of all the possible $X^t_{r,s}$ within the valid time frame for request r:

$$\bigvee_{t=t^a_r}^{t^d_r} \bigvee_{s \in \mathcal{S}} X^t_{r,s} = 1. \tag{5}$$

A feasible schedule for the multi-objective job scheduling problem exists if there's a satisfying assignment for the corresponding SAT problem. This establishes the multi-objective job scheduling problem as at least as complex as the SAT problem, implying the absence of an efficient polynomial-time solution. Addressing this complexity, we introduce a heuristic scheme (ABS), which is designed to schedule as many requests as possible within a feasible polynomial time frame.

C. Proposed ABS Scheme

For dense NR-V2X network, we proposed the ABS scheme which structured into three distinct phases: decision, queuing, and scheduling. The decision and queuing phases act as preprocessing stages. They are responsible for managing the incoming requests by first deciding which requests to process and then organizing them in a queue. This setup prepares the requests for the scheduling phase, where they are actually scheduled for dissemination. This structured approach streamlines the process, ensuring efficient handling of requests in the system.

Algorithm 1: Queuing Phase

Input: Set of requests $\mathcal{R}$, with priorities, popularity, and deadlines
Output: Set of ordered requests, O

Initialize the max-priority queue, Q.
for r ***in*** $\mathcal{R}$ **do**
  Enqueue r into Q
Initialize an empty list for ordered requests, O.
while Q *is not empty* **do**
  $r_1 \leftarrow$ Dequeue the top element from Q **if** Q *is not empty* **then**
    $r_2 \leftarrow$ Peek the next top element from Q. **if** *Priority*(r_1) > *Priority*(r_2) **then**
      Append r_1 to O;
    else if *Priority*(r_1) < *Priority*(r_2) **then**
      Append r_2 to O; Reinsert r_1 into Q;
    else
      if *Popularity*(r_1) > *Popularity*(r_2) **then**
        Append r_1 to O;
      else if *Popularity*(r_1) < *Popularity*(r_2) **then**
        Append r_2 to O; Reinsert r_1 into Q;
      else
        if *Deadline*(r_1) < *Deadline*(r_2) **then**
          Append r_1 to O;
        else
          Append r_2 to O; Reinsert r_1 into Q;
  else
    Append r_1 to O;

1) Decision Phase: In the decision phase, the RSU assesses whether the tile requests from vehicles need to be added to the scheduling queue. If the requested tile is not already scheduled for imminent broadcast or set to be broadcasted before the vehicle reaches the tile's position, the request is then pushed into the scheduling queue. This decision hinges on comparing the current time t with the broadcast time T^r_b of the tile request r. If T^r_b - t is less than T_{pos}, indicating the tile will be broadcasted in time, the request is not queued; otherwise, it is added to the queue for processing.

2) Queuing Phase: In the queuing phase mentioned in Algorithm 1, we employ a max-priority queue Q, to organize the tile requests based on a composite priority derived from layer priority, popularity, and deadline. Each new request is efficiently inserted into this queue according to its overall priority. Initially, the priority of requests is the primary factor for ordering. When requests have identical priorities, popularity and deadline serve as secondary criteria for sorting. Additionally, by manipulating these priorities and combining them into weighted ratios, we can develop various schedulers to optimize queue management further, tailoring it to different operational requirements or scenarios.

3) Scheduling Phase: In the scheduling phase mentioned in Algorithm 2, the RSU selects the request r_1 from the queue (ordered in descending order of priority in Algorithm 1) and determines its type. Different rules apply depending on the request type. Requests for osm layer (l_{osm}), deemed the most critical layer, receive the highest priority. The RSU prioritizes scheduling these l_{osm} requests, even if it means replacing a pcd request (l_{pcd}), to ensure at least basic navigation functionality of the vehicle. It searches for suitable slots between the request's arrival time and deadline, favoring empty slots or those with lower priority requests. For high-speed vehicles, the availability of candidate slots decreases, potentially leading to schedule misses if no suitable slot s is found. The pcd requests are scheduled into either empty spaces or lower-priority pcd slots, recognizing a potential pcd starvation. The focus, however, remains on minimizing osm misses to maintain essential navigation capabilities.

During the decision phase, each of the N requests undergoes a constant-time comparison to ascertain the necessity of queuing. In the queuing phase, a max-priority queue is employed for both sorting and inserting elements. This step introduces a time complexity of $O(NlogN)$ [16]. Following this, the scheduling phase requires, in the worst-case scenario, examining all possible time slots from the arrival time t^a to the deadline t^d for each request. Consequently, this results in a time complexity of $O((t^d - t^a + 1) \cdot N)$. Given that $(t^d - t^a + 1)$ is significantly smaller compared to N, the overarching time complexity of the algorithm is effectively $O(NlogN)$.

D. Demonstration of Considered Approaches

To provide a brief overview of ABS, consider requests from the vehicle, r_1 to r_8 which vary in priority, popularity, and deadline, reflected in a color spectrum from dark red (high priority) to light blue (low priority) as illustrated in Fig. 3.

Algorithm 2: Scheduling phase

Input: Request r with its properties
Output: Request r scheduled in slot s at time t

isPcdSlotForOsmFound ← False
isOsmSlotForOsmFound ← False
if $Type(r) = osm$ **then**
  for i ***in*** *range(a,d)* **do**
    if $X^t_{r,s_i} = l_{pcd}$ **then**
      $X^t_{r,s_i} \leftarrow 1$;
      isPcdSlotForOsmFound ← True
      $Break$;
  if $isPcdSlotForOsmFound$ = *False* **then**
    for i ***in*** *range(a,d)* **do**
      if $X^t_{r,s_i} = l_{osm}$ & $Popularity(r_i) < Popularity(r)$ **then**
        $S_i \leftarrow r$; $X^t_{r,s_i} \leftarrow 1$;
        isOsmSlotForOsmFound ← True
        $Break$;
  if $isOsmSlotForOsmFound$ = *False* & $isPcdSlotForOsmFound$ = *False* **then**
    Schedule miss for r.
if $Type(r) = pcd$ **then**
  for i ***in*** *range(a,d)* **do**
    if $X^t_{r,s_i} = l_{pcd}$ & $Popularity(r_i) < Popularity(r)$ **then**
      $S_i \leftarrow r$; $X^t_{r,s_i} \leftarrow 1$;
      isPcdSlotForOsmFound ← True
      $Break$;

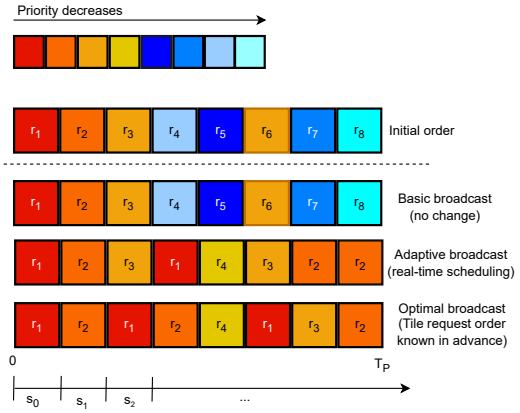


Fig. 3: Demonstration of considered approaches.

These are individual layer requests for an individual tile, which we assume for simplicity are divided in such as way that they can be stored in a single NR-V2X slot. Initially, in the pre-processing phase, the requests are broadcast in the order they are stored in the RSU, which may not follow any logical sequence, until the broadcast period T_P. In basic broadcast, this order remains unchanged. However, in adaptive broadcast, higher priority tiles are broadcast more frequently, phasing out less requested ones. The optimal broadcast, theoretically the most efficient, would require prior knowledge of request order to minimize schedule misses and TAT. One key distinction between adaptive and optimal broadcasting is the prevalence of high-priority tiles. Adaptive broadcasting features fewer high-priority tiles compared to the optimal approach, as the optimal pattern is not known beforehand. While adaptive broadcasting is not entirely optimal, it closely approximates the optimal and adapts in real-time to request patterns.

TABLE II: Delay comparison of different scheduling schemes.

	Basic	**Adaptive**	**Optimal**
Direct Hit	T_x+T_{p1}+T_{p2}	T_x+T_{p1}+T_{p2}	T_x+T_{p1}+T_{p2}
Hit with wait	T_x+T_{p1}+T_{p2}+T_b	T_x+T_{p1}+T_{p2}+T_b	NA
Scheduled Hit	NA	T_x+T_{p1}+T_{p2}+T_{pr}	NA
Miss	T_x+T_{p1}+T_{p2}+T_l	T_x+T_{p1}+T_{p2}+T_l	T_x+T_{p1}+T_{p2}+T_l

E. Comparison Approaches

Here, we compare the basic broadcast, optimal broadcast, and ABS. To the best of our knowledge, no such methods exist in the literature at the moment of writing this paper. Hence, the proposed heuristic solution has been compared with the optimal solution. The different delays considered for evaluation are specified in Table II.

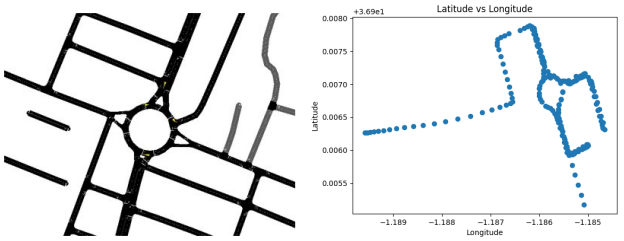
- **Basic Broadcast:** This conventional scheme involves broadcasting all tiles periodically in the same order, without any replacements in further cycles. A "direct hit" occurs when a tile is broadcast just as a vehicle requests it, ensuring immediate availability. However, the basic approach does not accommodate "scheduled hits" (tiles scheduled between arrival time and deadline), a feature present in the proposed ABS scheme. The "hit with wait" and "miss" scenarios are similar to the other approaches, with the former occurring when a tile is broadcast before the vehicle needs it, and the latter when a tile is unavailable before the vehicle reaches the location represented by the tile.
- **ABS:** This approach described in the Section IV-C, has an additional "Scheduled hit" category achieved after an additional processing delay.
- **Optimal Broadcast:** The optimal scheme described in Section IV-A in scheduling would be when we know in advance the order of requests. Which is why all the hits are "Direct Hit". Even when we know the order in advance we might not be able to schedule all requests hence the "miss".

V. Experiment Setup

A dense urban road segment of 1000×1000 m^2 (refer to Fig. 4a) has been set up in SUMO[3] which allows modeling of different traffic scenarios. The proposed ABS scheme thrives when there exists a pattern of non-uniformity in the tile requests. In our model same as in NR-V2X, a slot is a time segment for data transmission, lasting 1 ms in numerology 0 [14]. To evaluate this, the entire map (grid) has been divided into tiles of sizes 10×10, 20×20, 30×30, and 64×64. The broadcast period, T_P is defined as the time

[3]https://www.eclipse.org/sumo/

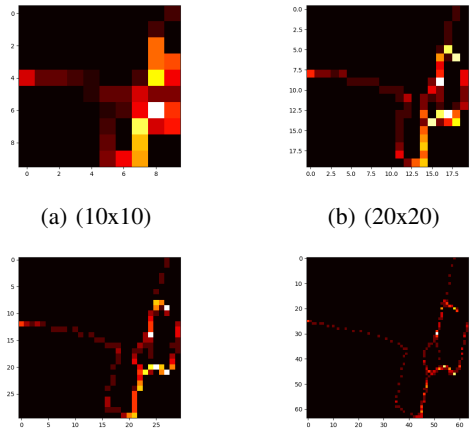
taken for the RSU to broadcast all the tiles. It is given by T_P = number of tiles × number of layers per tile. For an example, for a grid divided into 10×10 tiles having 2 layers each, the broadcast period would be 200 ms. Fig. 5 shows the heat map of the requests of various tiles of the grid by different vehicles for different tile sizes. As it is clear from the figure, the number of requests for certain tiles is higher when compared to some other tiles. A trajectory drawn (refer to Fig. 4b) of the vehicles show the fact that the tiles beyond the road-segment are seldom requested. This demand analysis conducted helps us to understand that there are tiles which are more popular than the others, while some tiles not being requested at all.



(a) An urban road segment scenario created in SUMO.

(b) Mapping of the vehicle trajectories.

Fig. 4: SUMO setup for HD Map tile request generation by vehicles in an urban road segment.



(a) (10x10)

(b) (20x20)

(c) (30x30)

(d) (64x64)

Fig. 5: Heat Map generated for different tile sizes.

A. Emulation setup

Our emulation setup built using Python ZMQ library[4], features a single RSU and a set of vehicles, each operating with dedicated threads. The RSU runs a broadcasting daemon thread and another thread for managing requests via worker threads. Vehicles have a thread for monitoring broadcasts from the RUS and another for requesting tiles if not found on the broadcast channel. They can run multiple threads by sending concurrent requests, containing their location and speed, which the RSU uses to determine tile numbers and decide on broadcasting schedule. The vehicle traffic data has been taken from SUMO (as mentioned in Table III), while the channel delays mentioned in Section III, have been extracted from Cellular-V2X module of NS-3[5] [17], to emulate V2I communication. This setup is available at [18].

TABLE III: Implementation setup parameters

Parameter	**Description**
System Type	NR-V2X Mode 1
Scenario	Urban intersection
Number of Vehicles	10
Total tile requests	1500
Speed of vehicles	$30 - 80$ KMPH
RSU data rate	100Mbps
Grid area considered	1000×1000 m^2 divided into 10×10, 20×20, 30×30, and 64×64 tiles.
Number of layers	2 layers each (.osm and .pcd)
Size of .osm layer	2KB
Size of .pcd layer	10KB
Size of tile request	2KB
Vehicle data rate	$3 - 5$ Mbps
Mobility model	SUMO based location
Broadcast cycle	200ms (10×10), 800ms (20×20), 1800ms (30×30) and 8192ms (64×64)

VI. Experiment Results and Analysis

These below performance metrics collectively help to get insights into the responsiveness and reliability of the proposed ABS under varying conditions.

- **Hit rate:** This is determined by the proportion of requests that are scheduled and fulfilled prior to their deadlines.
- **TAT:** This parameter represents the cumulative duration from the initial request of a tile to its successful reception by the vehicle, as observed on the broadcast channel.

A. Impact on Hit Rate

Fig. 6 shows the normalized hit rate for Basic, ABS, and Optimal broadcasting schemes. As the tile size decreases, the number of tiles to be broadcasted increases, thereby increasing the broadcast period, causing tiles to appear much later in the channel. This delayed appearance results in a higher frequency of ”misses” in the Basic approach. However, in case of ABS scheme, these tiles are scheduled before their respective deadlines, thereby significantly reducing misses and bringing performance closer to that of the Optimal broadcast.

B. Impact on TAT

Our emulation setup processed 1500 requests from 10 vehicles to assess TAT values. Basic Broadcast yields TAT values in the range of seconds, which is too slow for V2X applications. This inefficiency is due to it’s inability to alter the sequence of tile requests on the broadcast channel, leading to significantly higher TAT when a vehicle asks for a tile that has already been

[4]https://zeromq.org/languages/python/

[5]https://www.nsnam.org/

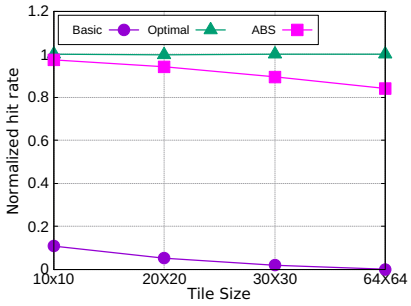


Fig. 6: Impact on Hit rate.

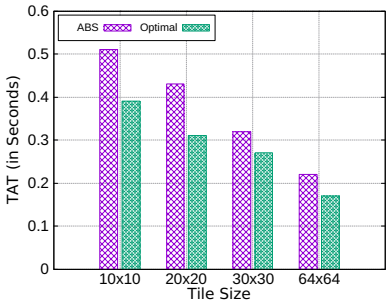


Fig. 7: Impact on TAT by varying tile sizes.

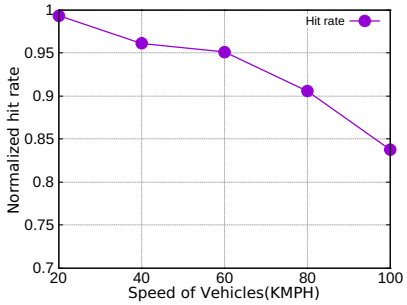


Fig. 8: Trend in hit rate with change in speeds of the vehicles.

broadcasted. In contrast, ABS achieved a significantly lower TAT of only 0.5 milliseconds per request on an average for all tile sizes When dealing with varying tile sizes, TAT decreases as the tile size reduces as shown in Fig. 7. Additionally, TAT with ABS is marginally higher compared to the Optimal Broadcast, which benefits from pre-existing request pattern knowledge, an aspect less viable in real-world scenarios.

C. Impact of Speed of Vehicles

In Fig. 8, we see that ABS is able to support below 10% schedule misses for speeds up to 80 KMPH. However, there is a dip in the hit rate as the speeds all the vehicles in the scenario go over 100 KMPH. This is primarily due to the requirement of delivering HD maps of size 1 GB per mile, as mentioned in [12], to all vehicles moving at speeds exceeding 100 KMPH. Such a scenario demands more bandwidth than what is currently supported in the setup.

VII. Conclusions and Future Work

We proposed ABS scheme for disseminating HD map data effectively to vehicles in dense NR V2X environments. An optimization model elevates the scheduling to an NP-Hard challenge, paralleling multi-objective scheduling schemes. Emulation results of ABS scheme show promise in reducing schedule misses as much as by 50% compared to the conventional periodic broadcast and minimizing turn around time per request. Future work includes enhancing ABS scheme using mini-slot scheduling to address challenges related to request starvation for lower priority requests. This technique involves distributing parts of different tiles within a single slot, offering a potential solution to improve request handling efficiency.

Acknowledgement

This work was supported by DST National Mission Interdisciplinary Cyber-Physical Systems (NM-ICPS), Technology Innovation Hub on "Autonomous Navigation and Data Acquisition Systems: TiHAN Foundations at Indian Institute of Technology (IIT) Hyderabad".

References

[1] Gamal Elghazaly, Raphaël Frank, Scott Harvey, and Stefan Safko, "High-definition maps: Comprehensive survey, challenges, and future perspectives", *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 527–550, 2023.

[2] "The evolution of the here hd live map at daimler", https://www.here.com/learn/blog/the-evolution-of-the-hd-live-map.

[3] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization", Tech. Rep. TR 102 863, ETSI, 2011.

[4] "Navigation data standard NDS", https://nds-association.org/.

[5] "Opendrive", https://www.asam.net/standards/detail/opendrive/.

[6] AECC, "Operational Behavior of a High Definition Map Application", Tech. Rep., Automative Edge Computing Consortium, 2020.

[7] Mario H. Castañeda Garcia, Alejandro Molina-Galan, Mate Boban, Javier Gozalvez, Baldomero Coll-Perales, Taylan Şahin, and Apostolos Kousaridas, "A tutorial on 5g nr v2x communications", *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021.

[8] Hakima Khelifi, Senlin Luo, Boubakr Nour, Akrem Sellami, Hassine Moungla, and Farid Naït-Abdesselam, "An optimized proactive caching scheme based on mobility prediction for vehicular networks", in *IEEE GLOBECOM*, 2018, pp. 1–6.

[9] Soohyun Park, Chanyoung Park, Soyi Jung, Minseok Choi, and Joongheon Kim, "Age-of-information aware contents caching and distribution for connected vehicles", *arXiv:2210.01536*, 2022.

[10] Fangfei Wang, Dong Guan, Long Zhao, and Kan Zheng, "Cooperative v2x for high definition map transmission based on vehicle mobility", in *IEEE VTC*, 2019, pp. 1–5.

[11] Xianzhe Xu, Shuai Gao, and Meixia Tao, "Distributed online caching for high-definition maps in autonomous driving systems", *IEEE Wireless Communications Letters*, vol. 10, no. 7, pp. 1390–1394, 2021.

[12] Qixia Hao, Jiaxin Zeng, Xiaobo Zhou, and Tie Qiu, "Freshness-aware high definition map caching with distributed mamab in internet of vehicles", in *Springer-Verlag*, 2022, p. 273–284.

[13] Athanasios Kanavos, Sokratis Barmpounakis, and Alexandros Kaloxylos, "An adaptive scheduling mechanism optimized for v2n communications over future cellular networks", in *MDPI Telecom*, 2023, vol. 4, pp. 378–392.

[14] Kenya Sato, Takahiro Koita, and Akira Fukuda, "Broadcasted location-aware data cache for vehicular application", *Springer EURASIP Journal on Embedded Systems*, pp. 1–11, 2007.

[15] Stephen A. Cook, "The complexity of theorem-proving procedures", in *ACM Symposium on Theory of Computing*, 1971, p. 151–158.

[16] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein, *Introduction to algorithms*, MIT press, 2022.

[17] Fabian Eckermann, Moritz Kahlert, and Christian Wietfeld, "Performance analysis of C-V2X mode 4 communication introducing an open-source C-V2X simulator", in *IEEE VTC*, 2019.

[18] "Hd map dissemination emulation setup", https://github.com/bugAssassin007/HDMapDisseminator/tree/master.