Data Offloading in Vehicular Networks

Project-I (CS47007) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Yarlagadda Srihas

(18CS10057)

Under the supervision of

Prof. Arobinda Gupta

Computer Science and Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2021-22

November 16, 2021

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 16, 2021 (Yarlagadda Srihas)

Place: Kharagpur (18CS10057)

COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Data Offloading in Vehicular Networks**" submitted by **Yarlagadda Srihas** (Roll No. 18CS10057) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2021-22.

Date: November 16, 2021
Place: Kharagpur

Prof. Arobinda Gupta
Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Contents

List of Figures

List of Tables

Abbreviations

RSU	**R**oad **S**ide **U**nit
V2V	**V**ehicle to **V**ehicle
V2I	**V**ehicle to **I**nfrastructure
DSRC	**D**edicated **S**hort **R**ange **C**ommunications
LTE-V2X	**L**ong **T**erm **E**volution **V**ehicle to **E**verything
OBU	**O**n **B**oard **U**nit
VANET	**V**ehicular **A**d-hoc **NET**work

1 Introduction

With increasing demand for data-hungry applications in vehicle infotainment systems, as well as the requirement of connected-vehicle based safety systems, it is important to ensure that the networks used to support such systems are managed efficiently. Cellular networks - which form the backbone of mobile internet - are likely to be utilized by these new applications developed for vehicles, leading to increased congestion. Hence, it is necessary to look at methods to reduce the direct usage of cellular networks for data delivery, and the anticipated adoption of vehicular network technology offers an avenue to solve this problem.

Vehicular networks typically include road-side units (RSUs) and vehicle on-board units (OBUs), which enable both vehicle-to-vehicle (V2V) connections and vehicle-to-infrastructure (V2I) connections. Heterogeneous vehicular networks, in particular, utilize various technologies like DSRC, LTE-V2X etc. in addition to cellular links to implement V2V and vehicle to RSU communication. The cellular data offloading problem for heterogeneous vehicular networks deals with the usage of these technologies to decrease the data transfer demand on cellular networks. Specifically, when data is required to be transferred between an external entity (like a server) and a vehicle that is part of a vehicular ad-hoc network (VANET), data offloading seeks to decrease the usage of cellular bandwidth by utilizing vehicle-to-RSU and inter-vehicle communication. The cellular channel may be used for sending/receiving control information, but the overall objective is to minimize its usage.

The data offloading problem can be classified into two major sub-problems - network selection and caching. The network selection problem refers to the problem of deciding whether to use the vehicular network or directly use the cellular network for retrieval/upload of data. The caching problem refers to the selection and identification of specified vehicles/RSUs as replicas for data, so that the data requests of certain vehicles can be served by a replica. Cache replacement policies are also a part of the caching problem, and must be addressed due to the finite availability of storage.

2 Related Works

The existing works on the data offloading problem can be categorized into two parts – solutions to the network selection problem and solutions to the caching problem.

Solutions to the Network Selection Problem: In solutions to the network selection problem, the choice of network is often based on the quality-of-service requirements of different applications. QoS requirements include acceptable delay, bandwidth requirements, tolerance for missing packets/errors etc. The selection decision is informed by the availability of the data in nearby vehicles and/or RSUs, as well as network factors like traffic and packet delay.

Push-and-track based algorithms [1], [2] are one class of solutions to the network selection problem. A set of seed vehicles are selected, either by a centralized controller or by a distributed mechanism, to disseminate the required data. This selection is made using data such as the location, speed and path of the vehicles, as well as anticipated proximity to RSUs or other vehicles. Some variants [3] use a contact graph to model this vehicular network, which is updated based on these parameters.

Other solutions include the application of a suitable clustering algorithm in order to identify gateways/cluster heads that will be responsible for the cellular data transfer on behalf of surrounding vehicles. Each vehicle sends/receives data from its nearest gateway instead of directly from the cellular network. The clustering algorithm may be implemented centrally [4] using the location details of the various vehicles, or may also be implemented in a distributed fashion [5] using broadcasting techniques.

Some solutions of the network selection problem focus on associating each network with a cost at the vehicle level, which enables choice of the cellular or vehicular network on the basis of connection status and various QoS requirements. This cost is used as a metric to determine which network a particular vehicle should use at a given time for data transfer.

Solutions to the Caching Problem: The caching problem focuses on storing frequently requested data in replicas, which are specified vehicles or RSUs. The choice of replicas is informed by proximity to requesting vehicles, grouping based on content similarity etc.

The solutions to the caching problem generally include tagging of various content using keywords, or other similar schemes to group content based on the interests of various consumers [6], [7]. The cache replicas are then chosen for these groups on the basis of various factors like proximity to the consumers of that type of content, and the data is directly dispatched to them using the cellular network. The replicas then disseminate the content to the subscribers.

Other solutions use probabilistic estimates to determine whether or not caching is required at a particular vehicle/RSU. Standard statistical distributions are used to model contact durations, and an optimization problem is formulated and solved. The preference for caching some types of files over others may also be decided on the basis of such estimates. Examples of such works include [8], [9].

It can be observed that works concerned with optimal network selection do not consider the effects of finite storage, cache replacement policies etc., while those concerned with caching do not consider the mobility information of the various vehicles. A caching policy that depends on the actual pattern of V2V contacts would likely lead to better replica placement, and thereby more data savings. Moreover, there is potential to consider network quality and congestion for caching decisions, rather than limiting its applicability to network selection.

3 The Data Offloading Problem

The problem consists of multiple vehicles travelling on their predetermined routes, and RSUs placed at fixed locations. The vehicles make requests for data items, which can be served (i) by a nearby vehicle that has the data item, (ii) by a nearby RSU that has spare capacity, or (iii) directly from the cellular network, through cellular base stations. The vehicles may pass through regions having varying cellular connectivity over their entire journey. Under this initial framework, we make the following assumptions:

1. RSUs are connected to the external world with a high speed backbone network, and hence transferring any data item to a RSU takes negligible time.

2. RSUs have infinite storage. Basically, this assumption, along with the previous one, means that all the data items can be seeded into the RSUs and any vehicle can download any data item from any RSU on its path. This is assumed for simplicity at this stage. However, all vehicles will still not be able to download everything from RSUs because of the constraints on the capacity, transmission rate, and contact time.

3. For each data item, a fixed number of vehicles are initially seeded with that data item at no cost. The choice of the vehicles will depend on the algorithm. Note that even after the initial seeding, a vehicle that is not a seed originally can get the data through other sources and act as a seed, and a vehicle that is a seed may no longer become a seed as it may choose to remove the data item from its cache based on its replacement policy if the cache is full. This is captured by the set of variables w_{ikt}.

4. A vehicle can act as a cache for a data item only if it has the complete data item. This is again done for simplicity at the beginning. However, note that a vehicle may still transfer only a part of a data item from another vehicle or RSU or cellular base station depending on the time of contact and transfer

policy. However, it cannot act as a source of the data item to any other vehicle unless it has the complete data item and caches it.

5. The storage for data items requested by a vehicle is separate from its cache and does not occupy any space in the cache, unless it is placed in the cache as per policy.

6. A vehicle requests for a specific data item only once. However, it can have more than one request open at the same time for distinct data items.

7. For any request, a vehicle only transfers the part of the data it does not have from another vehicle, RSU, or cellular base station (i.e., it does not get any part of the data more than once).

3.1 Problem Formulation

We first formally specify the inputs of the problem. Let the total time be divided into T time slots, numbered from 1 to T.

- **Network Quality:** Let $\mathcal{Q} = \{q_1^c, q_2^c, q_3^c, \ldots, q_k^c\}$ denote k possible cellular network quality any vehicle may be in. Let the date rate for transfer between the vehicle and the base station when the network quality is q_i^c be $d^c(q_i^c)$.

- **Vehicles:** Let $\mathcal{V} = \{v_1, v_2, v_3, \ldots, v_n\}$ be the set of vehicles, where $|\mathcal{V}| = n$. The caching capacity of a vehicle v_i is s_i^v. Let the network quality of a vehicle v_i at time t be denoted by $q_{i,t}^v$. The mean transfer rate between two vehicles is α^v. The vehicle v_i starts at time t_i^s, and reaches its destination at time t_i^f.

- **RSUs:** Let $\mathcal{R} = \{r_1, r_2, r_3, \ldots, r_m\}$ be the set of RSUs, where $|\mathcal{R}| = m$. The mean data transfer rate between a vehicle and RSU is β^r when the vehicle is in the range of an RSU, and the maximum number of vehicles that can connect to an RSU at any one time is γ^r.

- **Data Items:** Let $\mathcal{D} = \{d_1, d_2, d_3, \ldots, d_p\}$ be the set of data items that a vehicle can request for, where $|\mathcal{D}| = p$. For each $d_i \in \mathcal{D}$, the size of the data is l_i^d, and its lifetime is T_i^d.

- **Vehicle-to-Vehicle Contacts:** Let $X^{(t)}$ denote a symmetric matrix of dimension $n \times n$ where, for each position (i, j), element $X_{i,j}^{(t)} \in \{0, 1\}$ and $X_{i,j}^{(t)} = 1$ indicates that vehicles v_i and v_j are in contact range at time t.

- **Vehicle-to-RSU Contacts:** Let $Y^{(t)}$ denote a matrix of dimension $n \times m$ where, for each position (i, j), $Y_{i,j}^{(t)} \in \{0, 1\}$ and $Y_{i,j}^{(t)} = 1$ indicates that vehicle v_i and RSU r_j are in contact range at time t.

- **Requests:** Each vehicle v_i makes a sequence of requests for data items. Let $req_{i,j,t}^{v}$ denote a request made by the vehicle v_i for a data item d_j at time t. The request can be served by a combination of data transfer between v_i and another vehicle, an RSU, or a cellular base station.

For a given instance of the problem, we are required to find out the transfers each vehicle must make at each time instant to/from the other agents. Additionally, we also need to know which vehicles possess which data items fully at each time instant. Thus, the output of the problem is defined by the following sets of indicator variables.

- w_{ikt}, for all vehicles $v_i \in \mathcal{V}$, all data items $d_k \in \mathcal{D}$, and all time t, where $w_{ikt} = 1$ if vehicle v_i stores item d_k in its cache at time t, 0 otherwise.

- x_{ijkt}, for all vehicles $v_i, v_j \in \mathcal{V}, i \neq j$, all data items $d_k \in \mathcal{D}$, and all time t, where $x_{ijkt} = 1$ if vehicle v_i transfers part of data item d_k from vehicle v_j at time t, 0 otherwise.

- y_{ijkt}, for all vehicles $v_i \in \mathcal{V}$, all RSUs $r_j \in \mathcal{R}$, all data items $d_k \in \mathcal{D}$, and all time t, where $y_{ijkt} = 1$ if vehicle v_i transfers part of data item d_k from RSU r_j at time t, 0 otherwise.

- z_{ikt}, for all vehicles $v_i \in \mathcal{V}$, all data items $d_k \in \mathcal{D}$, and all time t, where $z_{ikt} = 1$ if vehicle v_i transfers part of data item d_k using the cellular network at time t, 0 otherwise.

The main goal of the problem is to minimize the access to the cellular network. However, a trivial solution for that is to not transfer anything (so no request is satisfied). Hence, we formulate the problem as maximizing the number of requests satisfied (requested data item downloaded in its entirety within time T) while minimizing the total amount of data (over all requests, satisfied or not) transmitted using the cellular network.

Let us define an indicator variable $I_{i,k,t}$ which is set to 1 if $req^v_{i,k,t}$ is satisfied, 0 otherwise. The request will be satisfied if the total data transferred by the vehicle v_i for data item d_k from any other vehicle, RSU, or the cellular network is equal to the size of d_k. The time within which $req^v_{i,k,t}$ has to be satisfied is given by $T^{req}_{i,k,t} = \min(t + T^d_k, T)$.

$$I_{i,k,t} = \begin{cases} 1 \text{ if } (\sum_{v_j \in \mathcal{V}} \sum_{\tau=t}^{T^{req}_{i,k,t}} x_{ijkt}) \times \alpha^v + (\sum_{r_j \in \mathcal{R}} \sum_{\tau=t}^{T^{req}_{i,k,t}} y_{ijkt}) \times \beta^r + \sum_{\tau=t}^{T^{req}_{i,k,t}} (z_{ikt} \times d^{cell}(q^v_{i,t})) = l^d_k \\ 0 \quad \text{otherwise} \end{cases}$$

Then the problem can be formulated as:

$$\text{maximize} \sum_{v_i \in \mathcal{V}} \sum_{d_k \in \mathcal{D}} \sum_{t=1}^{T} I_{i,k.t}$$

$$\text{minimize} \sum_{v_i \in \mathcal{V}} \sum_{d_k \in \mathcal{D}} \sum_{t=1}^{T} z_{ikt}$$

subject to the constraints:

- A vehicle v_i can transfer a data item d_k from another vehicle v_j only if they are in range and v_j caches the data.

$$x_{ijkt} = 1 \implies (X_{i,j}^{(t)} = 1 \wedge w_{jkt} = 1) \tag{1}$$

- A vehicle v_i can transfer a data item d_k from a RSU r_j only if they are in range.

$$y_{ijkt} = 1 \implies Y_{i,j}^{(t)} = 1 \tag{2}$$

- The number of vehicles that can transfer data from a RSU at any time t is less than or equal to γ^r.

$$\sum_{v_i \in \mathcal{V}} \sum_{d_k \in \mathcal{D}} y_{ijkt} \leq \gamma^r, \;\; \forall r_j \in R, \; \forall t \in T \tag{3}$$

- The total size of all data items cached in a vehicle v_i at any time t is less than or equal to s_i^v.

$$\sum_{d_k \in \mathcal{D}} (w_{ikt} \times l_k^d) \leq s_i^v, \;\; \forall v_i \in \mathcal{V}, \; \forall t \in T \tag{4}$$

- All transfers start after a vehicle's start time and finish before it reaches its destination.

$$x_{ijkt} = 0, y_{ijkt} = 0, z_{ijkt} = 0 \;\; \forall t < t_s^i, \; t > t_i^f \tag{5}$$

3.2 Model Simulation

The optimization problem proposed above is a mixed-integer problem, known to be NP-hard. The model was created in LP format and solved using the Gurobi mixed-integer optimizer (Simplex, branch-and-bound with cutting planes). It was

observed that, for a random input on average, either the time taken to arrive at an optimal solution was very long or the model was unable to run at all due to space constraints. The model size and observed solution time for different input sizes is summarized in Table 1.

TABLE 1: Variation in optimization model size and runtime

No. of vehicles	No. of constraints	Model file size (GB)	Runtime (s)
10	356,962	0.85	15295.85
20	1,394,008	3.8	Out-of-memory
30	2,947,816	8.7	Out-of-memory
No. of data items	**No. of constraints**	**Model file size (GB)**	**Runtime (s)**
1	356,962	0.85	15295.85
5	1,193,410	4.2	Out-of-memory
10	2,238,970	8.4	Out-of-memory
Distance (km)	**No. of constraints**	**Model file size (GB)**	**Runtime (s)**
5	175,300	0.22	716.46
10	356,962	0.85	15295.85
20	719,678	3.5	> 32000.00

Note: Unless otherwise mentioned, the default number of vehicles is 10, number of data items is 1 and distance is 10 km. Testbench: Lenovo ThinkPad-T480 (4-core, 16GB RAM)

Thus, the optimization model is infeasible for realistic input sizes, providing motivation for the development of a heuristic algorithm.

4 Algorithm

We make some additional simplifications to the problem. We consider that there is a single data item of length l_d, with data lifetime equal to the total time T. The time at which a vehicle opens a request for the data item is the time at which it starts moving (t_i^s). Every vehicle can receive the data item from an RSU in one go, and no vehicle downloads only a part of the item from an RSU. The following is a summary of the notations used in the following discussion:

- Let the set of known caches be denoted by $\mathcal{V}_{cache}$. This set is updated whenever a vehicle is designated to be a cache under the current allotment.
- For each RSU r_i, let the set of vehicles in range be denoted by ψ_t^i, and the set of vehicles actively receiving data by ϕ_t^i.
- The set of seed vehicles that have the data item from the start time $t = 0$ is denoted by $\mathcal{V}_{seed}$.
- Let the amount of the data held by a vehicle v_i at time t be denoted by $f_{i,t}$.
- In the score calculation, the weight given to data that can fully satisfy the request of a contacted vehicle is denoted by w_{sat}; the weight is w_{nsat} otherwise.

The general idea of the algorithm is to compute a score for each vehicle to rank the vehicles arriving at each RSU, on the basis of the amount of data they can transfer to other vehicles in the future. Consider the scenario where, say, 5 vehicles are in range of an RSU r_i at a particular time instant t, and none of them are already receiving data from the RSU. The method of score calculation is illustrated as follows: suppose that the score is being calculated for vehicle 1. The time t_0 at which the vehicle would fully receive the data item from the RSU is calculated (the vehicle is ignored if the full data cannot be transferred from the RSU), and over the interval $[t_0, T]$ we consider the contacts between vehicle 1 and the remaining vehicles given by the contact matrix $X^{(t)}$. If, for instance, vehicle 2 has already secured $l_d/2$ amount of data, it would require a further $l_d/2$ amount to satisfy its request. If the

total contact duration between vehicle 1 and 2 in the above time interval is sufficient to transfer $l_d/2$ amount of data, we add $w_{sat} * l_d/2$ to the score for vehicle 1; otherwise the score is incremented by the maximum possible data transfer weighted by w_{nsat}. This is repeated for the other vehicles $3 \dots 5$ to get the score of vehicle 1. While calculating the score, we only consider data transfers if the contacted vehicle is itself not a cache, and thus if any of the remaining vehicles has already secured the full data l_d, they do not contribute to the score calculation (vehicles $v_j \in \mathcal{V}_{cache}$ are not considered). Essentially, we prioritise vehicles that can i) help complete the most requests in the future and ii) transfer the most data to other non-cache vehicles.

The vehicle with the highest score, say v_{best}, is chosen to be a cache, and is added to list of transfers for the RSU ϕ_t^i. We then update the values of $f_{k,t}$ for all vehicles v_k that are contacted by v_{best} to reflect the data transfer up to the end time T. v_{best} is added to the list of known caches $\mathcal{V}_{cache}$.

Further, some of these vehicles v_k may become caches themselves. We keep track of these using the set $\mathcal{V}_{new}$ and apply the above procedure recursively. Thus, we obtain an exhaustive list of known caches $\mathcal{V}_{cache}$ at time instant t, accounting for all future vehicle-to-vehicle contacts.

Another vehicle from the remaining vehicles is then selected, and this procedure is repeated until the RSU reaches its capacity γ^r or there are no more vehicles left that are not already on the RSU transfer list ϕ_t^i. Doing this for all the RSUs over the full time interval yields the outputs - ϕ_t^i and $f_{j,t}$ for all t in $1 \dots T$, where ϕ_t^i gives the allocation of RSUs to vehicles and $f_{j,t}$ gives when vehicles receive their data item. If a vehicle does not receive the complete data item at T, then it is assumed to use the cellular data to complete its request.

The complete logic is split into three parts that are described with their accompanying pseudocode below. Algorithm 1 is used to calculate the score of a vehicle arriving at an RSU, in order to determine whether or not to transfer data to it. Algorithm 2 is used to update the storage of vehicles, given a set V_{new} of new caches to be added. Algorithm 3 utilizes algorithms 1 and 2 to allocate the RSUs to the vehicles.

Algorithm 1 Score Calculation - **score**(v_i, t_0):

1: **Inputs:** vehicle v_i, time t_0
2: **for** *each* vehicle $v_j \in \mathcal{V} \setminus \mathcal{V}_{cache}, v_j \neq v_i$ **do**
3: **if** $\sum_{\tau=t_0}^{T} X_{i,j}^{(\tau)} * \alpha^v \geq (l_d - \max_{\tau=t_0}^{T} f_{j,t})$ **then**
4: Add $w_{sat} * (l_d - (\max_{\tau=t_0}^{T} f_{j,t}))$ to *score*
5: **else**
6: Add $w_{nsat} * (\sum_{\tau=t_0}^{T} X_{i,j}^{(\tau)} * \alpha^v)$ to *score*
7: **end if**
8: **end for**
9: **return** *score*

Algorithm 1 Description: The loop runs for all non-cache vehicles contacted by v_i. The condition in line 3 checks whether the contact duration between v_i and v_j is sufficient to satisfy the request of v_j. The maximum amount of data that can be transferred is given a weight accordingly and added to the total score. The function returns the calculated score.

Algorithm 2 V2V Storage Update - **updateStorage**$(\mathcal{V}_{new}, \mathcal{V}_{cache}, f)$:

1: **Inputs:** set of new caches $\mathcal{V}_{new}$, set of known caches $\mathcal{V}_{cache}$, vehicle storage f
2: **for** *each* vehicle $v_j \in \mathcal{V}_{new}$ **do**
3: $t_0 \leftarrow \min(\tau)$ s.t $f_{j,\tau} = 1$
4: **for** *each* vehicle $v_k \in \mathcal{V} \setminus \{v_j\}$ **do**
5: **for** τ in $(t_0 + 1) \dots T$ **do**
6: $f_{k,\tau} \leftarrow \min(f_{k,\tau-1} + X_{j,k}^{(\tau)} * \alpha^v + old(f_{k,\tau} - f_{k,\tau-1}), l_d)$
7: **end for**
8: **if** $f_{k,T} = l_d \wedge v_k \notin \mathcal{V}_{cache}$ **then**
9: $\mathcal{V}_{new} \leftarrow \mathcal{V}_{new} \cup \{v_k\}$
10: **end if**
11: **end for**
12: $\mathcal{V}_{new} \leftarrow \mathcal{V}_{new} \setminus \{v_j\}$
13: $\mathcal{V}_{cache} \leftarrow \mathcal{V}_{cache} \cup \{v_j\}$
14: **end for**
15: **Updates:** set of known caches $\mathcal{V}_{cache}$, vehicle storage f

Algorithm 2 Description: A loop runs over the set V_{new} until it is empty. In line 3, the time t_0 at which the vehicle v_j completes its request is found. Then, lines 4-11 describe the process of updating the storage for each vehicle v_k contacted by v_j, from time t_0 till end time T. The data transferred from v_j to v_k at a particular

time instant τ is given by $X_{j,k}^{(\tau)} * \alpha^v$. The pre-existing difference $old(f_{k,\tau} - f_{k,\tau-1})$ signifies the data that v_k has already secured from other sources. Thus, to get the updated storage $f_{k,\tau}$ at time τ, these two quantities are added to $f_{k,\tau-1}$ in line 6.

In lines 8-9, we check if any of the contacted vehicles v_k has become a cache as a result of the new transfers, and add it to $\mathcal{V}_{new}$ if so. Finally, in lines 12-13 the sets $\mathcal{V}_{new}$ and $\mathcal{V}_{cache}$ are updated with the newly added cache v_j.

Algorithm 3 Contact based ranking

1: Initialize $f_{i,t}$ to 0 $\forall v_i \in \mathcal{V} \setminus \mathcal{V}_{seed}$ and $f_{j,t}$ to 1 $\forall v_j \in \mathcal{V}_{seed}, t \in [1, T]$
2: $\mathcal{V}_{new} \leftarrow \mathcal{V}_{seed}$
3: **updateStorage**$(\mathcal{V}_{new}, \mathcal{V}_{cache}, f)$
4: **for** t in $1 \dots T$ **do**
5: **for** *each* RSU r_i **do**
6: Update ψ_t^i based on RSU contact matrix $Y^{(t)}$
7: $\phi_t^i \leftarrow \phi_{t-1}^i$
8: **for** *each* vehicle $v_j \in \phi_t^i$ **do**
9: **if** $f_{j,t} = l_d$ **then**
10: $\phi_t^i \leftarrow \phi_t^i \setminus \{v_j\}$
11: **end if**
12: **end for**
13: **while** $(|\phi_t^i| < \gamma^r) \wedge (|\psi_t^i| - |\phi_t^i| > 0)$ **do**
14: $t_0 \leftarrow t + ceil[(l_d - f_{j,t})/\beta^r]$ for $v_j \in \psi_t^i \setminus \phi_t^i$
15: $v_{best} \leftarrow \text{argmax}_{v_j}\{\textbf{score}(v_j, t_0)\}$ over $(v_j \in \psi_t^i \setminus \phi_t^i$ and $Y_{i,j}^{t_0} = 1)$
16: $\phi_t^i \leftarrow \phi_t^i \cup \{v_{best}\}$
17: **for** τ in $(t+1) \dots T$ **do**
18: $f_{best,\tau} \leftarrow \min(f_{best,\tau-1} + \beta^r + old(f_{best,\tau} - f_{best,\tau-1}), l_d)$
19: **end for**
20: $\mathcal{V}_{new} \leftarrow \{v_{best}\}$
21: **updateStorage**$(\mathcal{V}_{new}, \mathcal{V}_{cache}, f)$
22: **end while**
23: **end for**
24: **end for**

Algorithm 3 Description: Lines 1-2 initialize the storage variables $f_{i,t}$ for all vehicles and the set $\mathcal{V}_{new}$ based on the given input $\mathcal{V}_{seed}$. **updateStorage**() is called in line 3 to account for vehicle-to-vehicle contacts from the seed vehicles, and $\mathcal{V}_{cache}, f$ are updated as a result.

In lines 4-24, a loop is run over all time instants and all RSUs. Line 6 gets the current set of vehicles in range of the RSU r_i. Lines 7-12 update the RSU transfer list to exclude vehicles that have completed their transfer. The loop in lines 13-22 runs until there are more vehicles in range of the RSU r_i that are not receiving data from it, and the RSU is not at capacity.

In line 14, the time t_0 at which a candidate vehicle would completely receive the data item from r_i is calculated. Over all such vehicles v_j where t_0 is actually within the contact duration of v_j and r_i, **score**() is called to calculate their respective scores. The best vehicle is added to the transfer list in line 16. The loop in lines 17-19 updates the storage of the chosen vehicle v_best to reflect the data received from r_i. Finally, **updateStorage**() is called with $\mathcal{V}_{new}$ initialized to the singleton set $\{v_{best}\}$, thereby updating the $\mathcal{V}_{cache}$ with the newly added cache(s) and the storage f of all contacted vehicles.

4.1 An Illustrative Example

We consider a complete example to illustrate the working of the algorithm. We consider all vehicles following a straight path of length 1km, and a single RSU r_i placed at the 500m mark, with coverage from 200m to 800m. The size of the data item is 330Mb (41.25 MB). The constant values are the same as described in Table 3. We impose a limit on the number of simultaneous transfers - a vehicle can transfer data to/from only a single agent at a time, with preference given to RSUs.

Let there be 5 vehicles in the set $\mathcal{V}$, each denoted by v_i, $i \in [1, 5]$. There is a single seed vehicle v_4, and thus $\mathcal{V}_{seed}$ is initialized to $\{v_4\}$. The starting times and speeds of the vehicles are summarized in Table 2.

TABLE 2: Speeds and starting times of the vehicles

Vehicle	Speed (m/s)	Starting time (s)
v_1	10	0
v_2	12	0
v_3	14	5
v_4	16	15
v_5	18	20

When the algorithm is run, the storage of all vehicles contacted by the seed vehicle v_4 is updated, and v_4 is added to $\mathcal{V}_{cache}$. The vehicles whose storage is updated are: v_1, v_2 and v_5. Following the initialization, the first vehicle that reaches 200m (the beginning of RSU range) is v_2, at time instant $t = 18$. The score computed for v_2 at this instant is 7, corresponding to the data it can transfer to v_3 after it becomes the cache (it is also in contact with v_5, but this overlaps with v_3). Note, during the calculation of v_2's score, contacts with v_4 are omitted as v_4 is already a known cache. As the RSU not serving any vehicles yet, v_2 is added to the transfer list ϕ_t^i and to $\mathcal{V}_{cache}$. The storage f of v_3 is updated to reflect the data transfer from v_4.

Then, two vehicles arrive within RSU range simultaneously at instant $t = 21$, namely vehicles v_3 and v_1. The score computed for v_3 is 19, and for v_1 is 1, and therefore the higher scoring v_3 is chosen to be the cache. v_3 is added to the transfer list and $\mathcal{V}_{cache}$, and the storage f of the vehicles contacted by v_3 (v_4 only) is updated. The RSU is now at its capacity, so v_1 cannot be accommodated.

There is no other vehicle that can arrive at the RSU at a later time instant such that the RSU is available and the vehicle can receive the full data item from it. Therefore, the remaining iterations pass without any more updates to $\mathcal{V}_{cache}$ or the storage f. At time instants $t = 50$ and $t = 53$, the vehicles v_4 and v_3 respectively are removed from the transfer list ϕ_t^i as they complete their request.

The algorithm terminates with $\mathcal{V}_{cache} = \{v_1, v_3, v_4\}$. The final storage of the vehicles is $[33, 330, 330, 330, 23]$ in order, and the amount of data remaining is 604 Mb (36%). The change in storage with time is plotted in Figure 1.

FIGURE 1: Vehicle storage v/s time instants

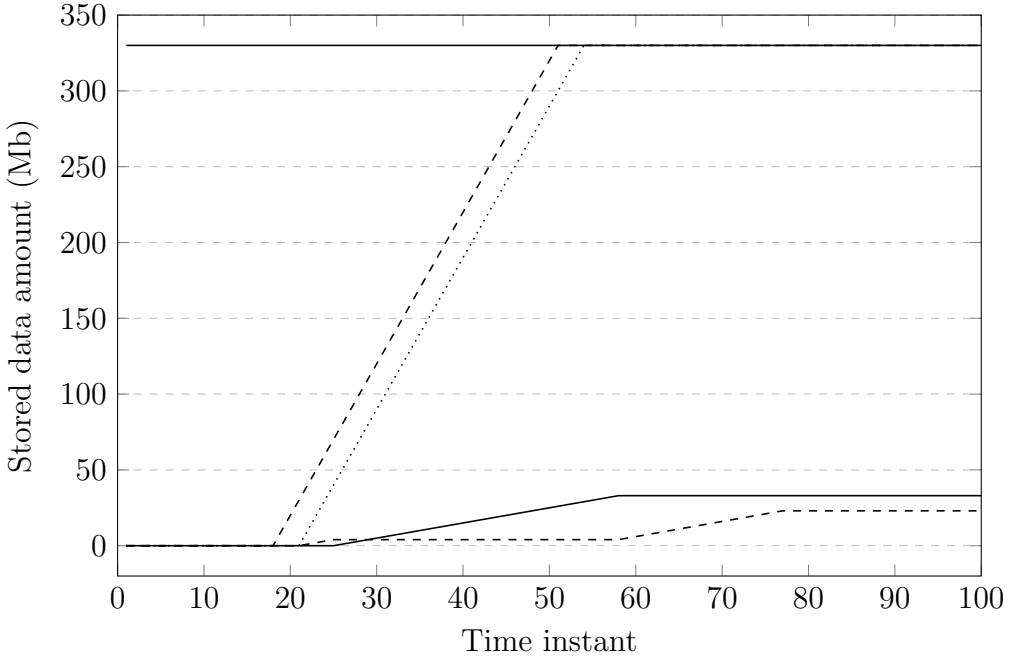
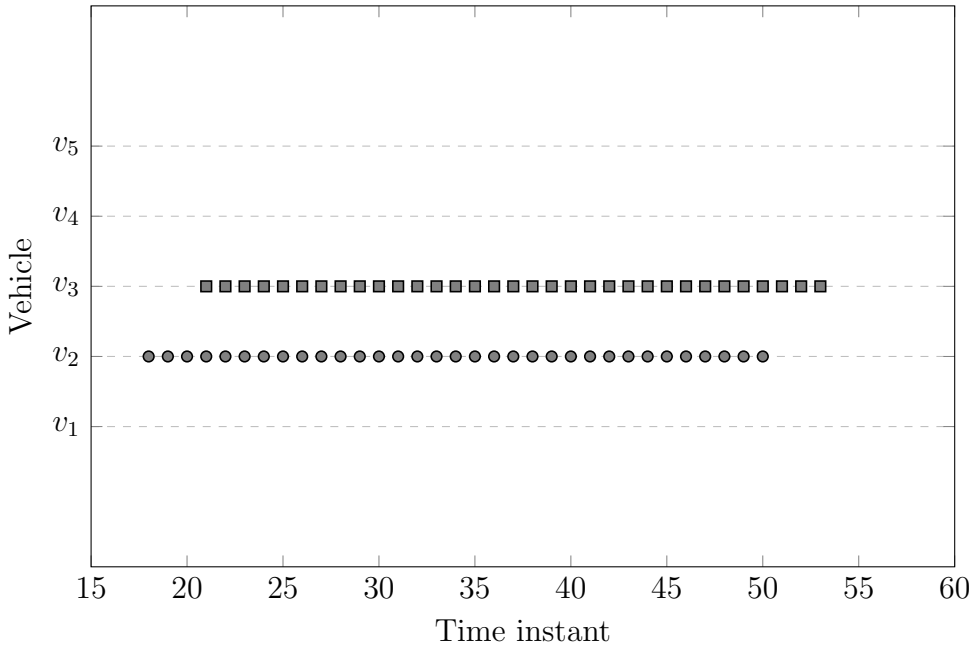


FIGURE 2: RSU allocation



5 Results

We have evaluated the performance of the algorithm by simulating over some initial scenarios. In all the inputs considered, the path taken by each vehicle was a straight track of length 10km. Each input test case was run for two values of $|\mathcal{V}|$ (number of vehicles), 20 and 50. Random values between 8-20 m/s (28.8 to 72 km/h) were used for the speed of each vehicle. The constant values used are summarized in Table 3 - they have been implemented as tunable parameters. There is a limit on the number of simultaneous transfers - a vehicle can transfer data to/from only a single agent at a time, with preference given to RSUs. The remaining parameters like number of RSUs, positions of the RSU and total vehicular contact duration were varied across test cases.

TABLE 3: Values of constants used

Quantity	Value
RSU range	300m
γ^r	2
β^r	10Mbps
Vehicle-to-vehicle range	100m
α^v	1Mbps
w_{sat}	1.5
w_{nsat}	1
l_d	300Mb

Number of RSUs: The algorithm was run with two different densities of RSU placements. In the case of 20 vehicles, 1 RSU (placed at 5000m) was considered for low density, and 2 RSUs (placed at 2000 and 6000m) were considered for high density. A single seed vehicle (v_1) was considered, resulting in a total demand of 5700. On the other hand, for 50 vehicles, 1 RSU (placed at 5000m) and 3 RSUs (placed at 2000, 4000 and 6000m) were considered for the two densities. Two seed vehicles were considered out of the 50 vehicles (v_{16}, v_{36}). The range of starting times for the vehicles was between 0 and 30 seconds, resulting in a fairly high V2V contact time as well as increased RSU competition. The data transferred through

RSU-to-vehicle and vehicle-to-vehicle communication is plotted in Figure 3, and the number of requests satisfied is summarized in Table 4.

FIGURE 3: Data transferred v/s number of RSUs

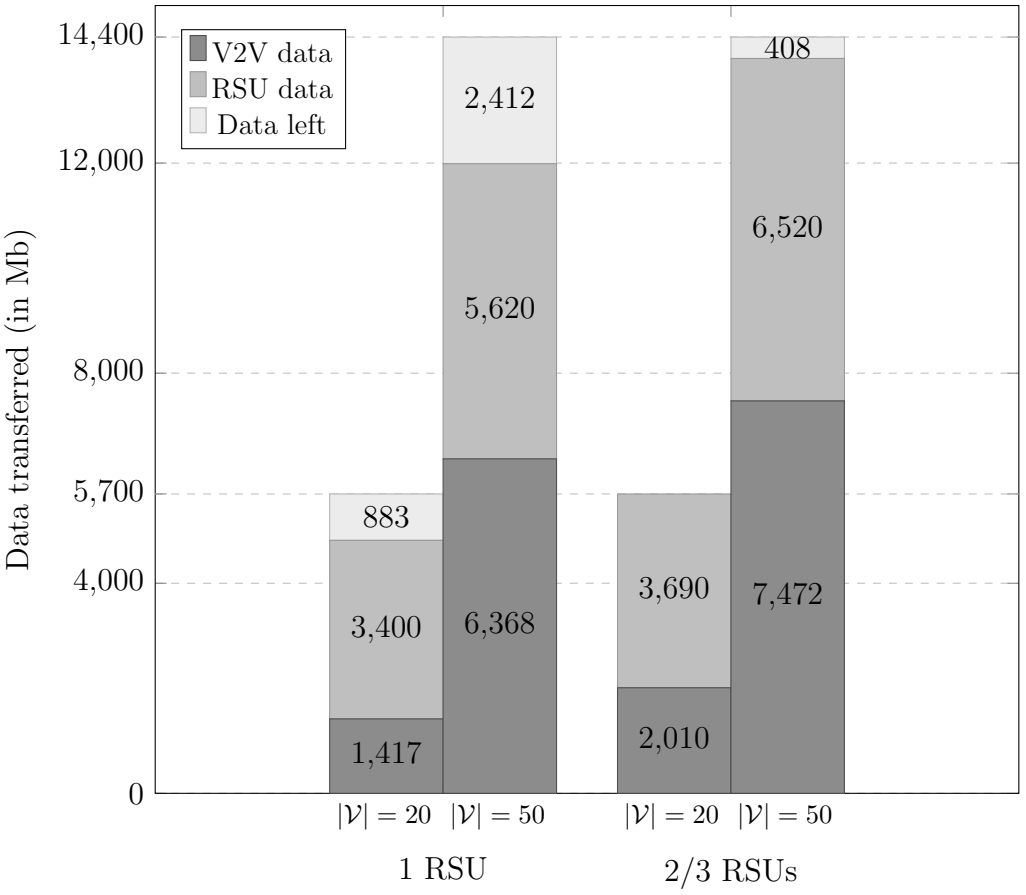


TABLE 4: Number of requests satisfied (varying RSUs)

Test Case	Requests satisfied (incl. $\mathcal{V}_{seed}$)	Data (Mb)
20 vehicles, 1 RSU	17	883
20 vehicles, 2 RSUs	20	0
50 vehicles, 1 RSU	38	2412
50 vehicles, 3 RSUs	48	408

As expected, increasing the number of RSUs corresponds to a significant increase in the amount of data transferred, and requests satisfied. The maximum percentage of data remaining was 16.5%, with all requests being satisfied in the case of 20

vehicles and 2 RSUs. Interestingly, there is also an increase in the V2V data transfer when the number of RSUs is increased, implying that the selected cache vehicles are indeed disseminating more information to other vehicles.

Duration of V2V contacts: The algorithm is now run for different amounts of vehicle-to-vehicle contacts. In all instances, a single RSU is placed at 5000m. In the low contacts case, the starting times of the vehicles are sampled from the interval [0,300] seconds, leading to longer gaps between vehicles. In the high contacts case, the starting times are in the range [0,30] seconds. As before, a single seed vehicle ($v1$) was taken in the 20 vehicles case, and two seed vehicles (v_{16}, v_{36}) were taken in the 50 vehicles case. The total V2V contact duration (including overlaps) as a result of varying the start times changed from 7054s to 11776s in the case of 20 vehicles, and 35529s to 92082s in the case of 50 vehicles. The results are summarized in Figure 4 and Table 5.

FIGURE 4: Data transferred v/s amount of contact

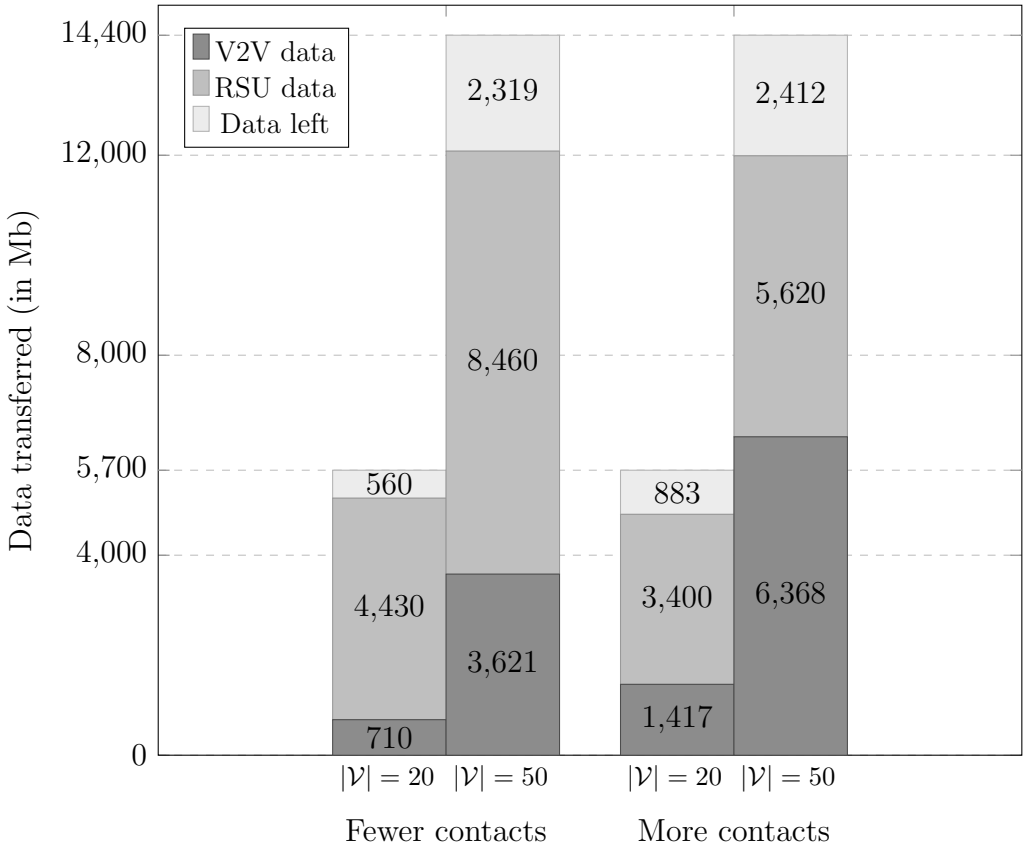


TABLE 5: Number of requests satisfied (varying contacts)

Test Case	Requests satisfied (incl. $\mathcal{V}_{seed}$)	Data (Mb)
20 vehicles, fewer contacts	18	560
20 vehicles, more contacts	17	883
50 vehicles, fewer contacts	37	2319
50 vehicles, more contacts	38	2412

With this set of inputs, it was observed that increasing vehicular contact time actually led to more data remaining in the end, seeming counter-intuitive at first glance. However, it can be noticed that the RSU transfers decrease significantly in the case where there are more vehicular contacts, this is due to more vehicles competing for the RSU simultaneously rather than utilizing it at disjoint intervals. As the RSU transfer speed is much higher than vehicle-to-vehicle transfer speed, the increased V2V contact duration is unable to compensate for the loss in RSU transfer. A similar trend is observed in the number of requests satisfied, excepting the 50 vehicles case where one additional request is satisfied despite less data transfer (likely due to the specific contact pattern in the input).

6 Plan of Work

We have proposed an initial version of the algorithm, and done simulations on some preliminary scenarios. In the next phase, we plan to do the following:

- Simulate the basic algorithm with a wider range of inputs (more vehicles/RSUs, varied paths). This, in addition to testing different parameter values (like w_{sat}, w_{nsat}), can help us understand the performance and scalability of the basic algorithm better.

- Generalize the algorithm to deal with more than one data item and multiple requests from each vehicle.

- Allow for cellular downloads during the vehicle's journey. The proposed algorithm deals with the allocation of RSUs to vehicles to disseminate data, but it may be beneficial to use the cellular network to seed a vehicle early in its journey if it can contribute more to other vehicles on its path.

- Use V2V contact details to determine which vehicle to download from. Under the current scheme, V2V data transfers occur in a first come, first serve manner. The exact V2V contact patterns can be utilized to enable a contact-aware choice of vehicular transfers, with more potential benefits.

- Incorporate incomplete downloads of data items from RSUs, and choose the optimal vehicles for such transfers. Potential candidates for such vehicles are those that can receive the rest of the data item, and thereby become caches themselves, promptly after leaving the RSU range.

- Test the algorithm with city-map based traffic simulations. Using realistic paths, vehicle movement patterns and traffic rates would provide a more accurate picture of the algorithm's performance, and help identify areas of improvement.

References

[1] John Whitbeck, Yoann Lopez, Jérémie Leguay, Vania Conan, and Marcelo Dias De Amorim. Push-and-track: Saving infrastructure bandwidth through opportunistic forwarding. *Pervasive and Mobile Computing*, 8(5):682–697, 2012.

[2] Filippo Rebecchi, Marcelo Dias de Amorim, and Vania Conan. Droid: Adapting to individual mobility pays off in mobile data offloading. In *2014 IFIP Networking Conference*, pages 1–9, 2014.

[3] Hong Yao, Deze Zeng, Huawei Huang, Song Guo, Ahmed Barnawi, and Ivan Stojmenovic. Opportunistic offloading of deadline-constrained bulk cellular traffic in vehicular dtns. *IEEE Transactions on Computers*, 64:1–1, 12 2015.

[4] Shucong Jia, Sizhe Hao, Xinyu Gu, and Lin Zhang. Analyzing and relieving the impact of fcd traffic in lte-vanet heterogeneous network. In *2014 21st International Conference on Telecommunications (ICT)*, pages 88–92, 2014.

[5] Abderrahim Benslimane, Tarik Taleb, and Rajarajan Sivaraj. Dynamic clustering-based adaptive mobile gateway management in integrated vanet — 3g heterogeneous wireless networks. *IEEE Journal on Selected Areas in Communications*, 29(3):559–570, 2011.

[6] Yong Li, Depeng Jin, Zhaocheng Wang, Lieguang Zeng, and Sheng Chen. Coding or not: Optimal mobile data offloading in opportunistic vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):318–333, 2014.

[7] Xiangming Zhu, Yong Li, Depeng Jin, and Jianhua Lu. Contact-aware optimal resource allocation for mobile data offloading in opportunistic vehicular networks. *IEEE Transactions on Vehicular Technology*, 66(8):7384–7399, 2017.

[8] Luigi Vigneri, Thrasyvoulos Spyropoulos, and Chadi Barakat. Storage on wheels: Offloading popular contents through a vehicular cloud. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, 2016.

[9] Weicheng Zhao, Yajuan Qin, Deyun Gao, Chuan Heng Foh, and Han-Chieh Chao. An efficient cache strategy in information centric networking vehicle-to-vehicle scenario. *IEEE Access*, 5:12657–12667, 2017.