Roadcast: A Popularity Aware Content Sharing Scheme in VANETs [*]

Yang Zhang *yangzhan@cse.psu.edu*
Jing Zhao *jizhao@cse.psu.edu*
Guohong Cao *gcao@cse.psu.edu*
Department of Computer Science & Engineering,
The Pennsylvania State University,
University Park, PA, USA

Content sharing through vehicle-to-vehicle communication can help people find their interested content on the road. In VANETs, due to limited contact duration time and the unreliable wireless connection, a vehicle can only get the useful data when it meets the vehicle which has the exactly matching data. However, the probability of such cases is very low. To improve the performance of content sharing in intermittently connected VANETs, we propose a novel P2P content sharing scheme called Roadcast. Roadcast relaxes user's query requirement a little bit so that each user can have more chances to get the requested content quickly. Furthermore, Roadcast ensures popular data is more likely to be shared with other vehicles so that the performance of overall query delay can be improved. Roadcast consists of two components called popularity aware content retrieval and popularity aware data replacement. The popularity aware content retrieval scheme makes use of Information Retrieval (IR) techniques to find the most relevant data towards user's query, but significantly different from IR techniques by taking the data popularity factor into consideration. The popularity aware data replacement algorithm ensures that the density of different data is proportional to the square-root of their popularity in the system steady state, which firmly obeys the optimal "square-root" *replication rule [6]. Results based on real city map and real traffic model show that Roadcast outperforms other content sharing schemes in VANETs.*

I. Introduction

The proliferation of low-cost wireless connectivity, combined with the growth of distributed peer-to-peer cooperative systems, is transforming the next-generation vehicular networks. With wireless technology, it is possible to deliver digital content from roadside infrastructure to drivers and passengers inside moving vehicles [18,25,31,33]. With the support of peer-to-peer wireless communication, content can be shared among vehicles beyond the infrastructure coverage [11,21,24]. Supporting content delivery and sharing in vehicular ad hoc networks (VANETs) can greatly benefit our daily life. For example, information about road hazards, traffic jams, and emergency stops can be used to improve traffic safety and efficiency. Passengers or drivers inside vehicles can get entertainment or local information such as MP3 music, sale advertisement, restaurant recommendations or videos of upcoming attractions.

Most existing research focuses on various solutions to disseminate some data to other vehicles [8, 11, 18, 21]. Another important problem is to efficiently find the requested data/content using VANETs. Currently, a user in the VANET can only get his/her interested data opportunistically, i.e., it gets the data only when it meets another vehicle which happens to have the requested data [11, 18, 21]. Obviously, such chances are very low in VANETs. Although service discovery techniques [12,27] are widely used in peer-to-peer networks and wireless ad hoc networks, it is difficult to apply them to VANETs. This is because VANETs may be sparsely connected [28,33], especially at night or at rural areas, and hence the delay and communication overhead of finding the requested data in VANET is much higher.

In this paper, we propose a novel content sharing scheme (called Roadcast) for VANETs. The motivation of the popularity-aware content sharing is as follows. If a vehicle requests a popular data which is densely disseminated in the network, it may take much shorter time than requesting a rare data, because the chance of meeting one vehicle that has the popu-

[*]A preliminary version [32] of the paper appeared in IEEE ICDCS'09. This work was supported in part by the National Science Foundation under grant number CNS-0721479.

lar data is much higher. In the opportunistic and unreliable VANET, we can expect that users are more willing to get data which roughly matches their interest with shorter delay than taking a longer delay (and the risk of not getting it) to get the perfectly matching data. As a result, it is desirable to give more opportunities to deliver the data with higher popularity. Getting popular data can also satisfy the neighboring node's query and may serve more vehicles in the future. Thus, we need to balance two objectives: *matching users' query* and *increasing data accessibility in the future*.

Roadcast achieves these objectives with two techniques: *popularity aware content retrieval* and *popularity aware data replacement*. First, the popularity aware content retrieval scheme makes use of information retrieval (IR) techniques to find the relevant data towards users' queries. Different from IR techniques, we consider the popularity factor of the data and re-rank the relevance of the data to queries, and ensure more popular data is more likely to be shared with other vehicles. Second, in Roadcast, the downloaded data is stored as replica which can be shared with other Roadcast users. When the local memory is full, some data has to be replaced. The proposed data replacement algorithm ensures that popular data has more, while not too many, replicas in the network. Our analysis shows that with the proposed data replacement algorithm, the densities of different data are proportional to the square-root of their popularity in the system steady state, which firmly obeys the optimal *"square-root" replication rule* [6]. Simulation results show that the proposed popularity aware content sharing solutions can reduce the data access delay while satisfying the user requirements.

The rest of this paper is organized as follows. Section II presents the related work. Then in Section III, we describe the proposed popularity aware content retrieval scheme, which is the first part of Roadcast. The second part of Roadcast, the popularity aware data replacement algorithm that can be used to achieve the optimal data allocation, is introduced in Section IV. Performance evaluations are shown in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

II.A. Vehicular Networks

Vehicular networks represent an interesting application scenario not only for traffic safety and efficiency but also for more commercial applications and entertainment support such as content sharing [10], peer-to-peer marketing [22], and urban data collecting [15, 20]. Most vehicular network researches have focused on routing issues. MDDV [29] provides a routing framework that exploits geographic forwarding to the destination region. VADD [33] and TBD [16] study how to choose the best routing path based on the traffic and trajectory information. Maxprop [4] determines packet delivery/drop order when node contact duration is not long enough to delivery all the packets. Zhao *et al.* [34] introduce data pouring and buffering techniques to disseminate data along the roads. All these works assume the content consumer is known beforehand so that the sender can route the content to its destination. Our work studies content sharing that is different from routing. In content sharing, each vehicle queries the useful data from its encountered neighbors, and the focus is how to retrieve and buffer the most suitable data from neighboring vehicles.

II.B. Content Retrieval

Recently, there has been increasing interest in content retrieval through intermittent contact opportunities in vehicular networks. [30] and [11] focus on a low power, low connectivity setting, where vehicles in adjacent lanes exchange information as they pass through one another. Guo *et al.* [11] further discusses how to retrieve interested data at particular region and particular time. They only study content retrieval in a small area. Our work, however, investigates content retrieval and sharing at a much larger scale. Lee *et al.* [21] and Johnson *et al.* [18] improve content retrieval by using randomized network coding. The data is cut into small blocks and encoded before being injected into the network. After enough number of blocks are collected, the original data can be recovered. All these works require exact match between user query and data. In Roadcast, different techniques are used to approximately match user query and data. We study how to efficiently sharing content with future encountered vehicles based on local information.

II.C. Data Replacement

Studies in data replacement start from cache replacement. Cao and Irani [5] have studied several replacement algorithms such as Least-Recently-Used (LRU), Least-Frequently-Used (LFU), Lowest Relative Value (LRV) for web cache, and have found none of the existing algorithms address the network cost concerns. They propose the GreedyDual-Size algorithm so that cache replacement considers both the variability in data size and the retrieval cost. Later, Jin and Bestavros [17] improve the GreedyDual-Size algorithm by adding the popularity factor. However, all these works are based on a centralized environment, where the web cache server collects information on the data access pattern to make better cache replacement decisions. The data replacement in Roadcast differs from the existing works in that it is a distributed replacement algorithm and it aims to optimize the network-wide content retrieval delay. Vehicles perform local replacement decisions based on their own knowledge. The collective behavior of all vehicles achieves a global square-root [6] replication allocation to provide better content access.

III. Popularity Aware Content Retrieval

III.A. Overview

In our popularity aware content retrieval, we should consider two important characteristics when vehicles request and retrieve content from the encountered vehicles.

1. Relevance between content and query.
   When one query is issued, it can generally be served by multiple data with different degree of relevance to the query. Many users do not necessarily require to get the exact matching content, e.g. John Lennon's song called Imagine. Instead they may only roughly describe their interested content at a coarse level, and hence they would be satisfied with any content close to their keyword query descriptor, e.g. any John Lennon's songs, or any MP3 rock music.

2. Tradeoff between content relevance and access delay.
   A VANET is generally known as an intermittently connected network, where the network connectivity is opportunistic and the connection duration is short and unreliable. Users can only query their encountered vehicles for data. Therefore, the delay to obtain the perfectly matching content is long. However, if the user requests can be relaxed a little bit, it may take much shorter time for the user to get the satisfied content. Thus, there is a trade-off between getting less interested data (but still nice to have) with better chances or getting perfectly matching data with less chances and longer delay.

Considering these characteristics, we give more opportunities to the content with higher popularity. That is, when one vehicle receives a content request from a neighboring vehicle, it returns the most popular content that is relevant to the request. The returned content can satisfy the neighboring vehicle's request, and serve more vehicles in the future. Thus, delivering more popular content contributes more to the content accessibility from the network perspective. In summary, when Roadcast chooses a data to deliver from one vehicle to another, the goal is to maximize and balance the following two aspects:

- Matching users' query.
- Increasing data accessibility in the future.

Different from other peer-to-peer content delivery and sharing schemes, the decision on which data to deliver in Roadcast considers the client's current interest and the overall demand in the network. A receiver is not only a content consumer, but also a producer which shares the content to others in the future. Therefore, the data retrieval considers not only serving the current receiver but also potentially serving more users in the future.

In Roadcast we extend the classical Information Retrieval (IR) algorithm to realize the popularity aware content retrieval in VANETs. Searching data items based on keywords has been extensively studied in the IR community [19,26]. However, their solutions are centralized and data accessibility is not an issue in their environments. In Roadcast, the basic idea is to leverage the most popular and well-studied IR algorithm, Vector Space Model (VSM), to find the data that matches users' query, and also consider data popularity as an important factor. Thus, Roadcast may not always deliver the best matching data for a reply, instead it delivers a less matching data, but more popular. Therefore more popular data is given more opportunities to be shared with others. This makes Roadcast very efficient at obtaining popular content. For less popular data, the delay may be longer. To address this problem, in Section IV, we propose techniques to keep some copies of less popular data in the network to optimize the overall performance.

In the following, we first describe how to use VSM to find the data that matches users' query, and then enhance the solution by considering data popularity.

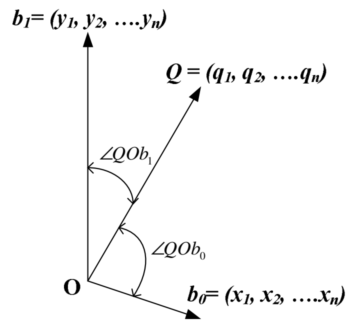
III.B. Matching queries based on VSM

Both data and query can be presented and indexed with resource representation techniques such as RDF (i.e., Resource Description Framework [13]) or WSDL (i.e., Web Services Description Language

[14]) based on specific keyword attributes. In Roadcast we also assume queries are keyword based. Users enter a sequence of keywords into the Roadcast system to describe the data item they want to retrieve, such as "MP3 music rock John Lennon Imagine".

To support complex data description we assume each data item is associated with multiple *tags* as the meta-data description in Roadcast. For example, a MP3 file of John Lennon's song Imagine is attached with a tag like "MP3 / music / rock / UK / 1970's/ John Lennon / Imagine". The tag of a data can be obtained from external sources and pre-loaded in Roadcast, or added/edited by Roadcast users.

Next, we describe how to use Vector Space Model (VSM) to find the data that matches users' query. Suppose there are m data items in a vehicle, and each data item is associated with some keywords as *tags* (as shown in Table 1). Let n denote the total number of terms that can be used in the tag and query vocabulary. Then each data item b_i can be represented by a binary vector in the n-dimensional space, say $\overrightarrow{b_i}$, whose entry indicates the presence or absence of one particular term in the tags of data item b_i. The entry is "0" if the term does not occur in the data tag, and "1" otherwise. In this way, the data items in the vehicle can be represented by a $m \times n$ binary matrix, where every row represents one data item (shown in Table 2). Similarly, a query can also be thought as a vector in the same n-dimensional space. For example, query "MP3 / music / John Lennon / Imagine" can be interpreted as a n-dimensional vector $(1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, ...)$. Thus, content retrieval becomes a matter of finding the data vectors in the space that are closest to the query vector.



$b_1 = (y_1, y_2, \ldots y_n)$
$Q = (q_1, q_2, \ldots q_n)$
$\angle QOb_1$
$\angle QOb_0$
O
$b_0 = (x_1, x_2, \ldots x_n)$

Figure 1: Similarity of vectors

To answer a query, the data items are ranked according to the similarity between the data vector and the query vector, and the data item with the highest similarity will be returned. A common measure of the similarity between two binary vectors with the same dimension is to calculate the number of their overlapped "1"s. If one data vector has more common "1"s with the query vector, its data has higher similarity with the query. However, this similarity comparison method has some bias since the data items with more terms tend to be ranked higher than those with fewer terms. Therefore, the number of terms that appears in the data term vector should be normalized when calculating the similarity.

We use the angle of two vectors to represent their similarity, which is able to remove the bias due to the number of occurrent terms. If two vectors have a smaller angle between them, they are more similar. To simplify the computation, the angle of two vectors can be transformed to the cosine value of the angle. Formally, as shown in Figure 1, given the n-dimensional term vector of a query Q, $\overrightarrow{Q} = (q_1, q_2, \cdots, q_n)$ and two data items, $\overrightarrow{b_0} = (x_1, x_2, \cdots, x_n)$ and $\overrightarrow{b_1} = (y_1, y_2, \cdots, y_n)$, the similarity between query Q and two data items b_0, b_1 can be defined in Equation 1.

$$\begin{aligned} cos(\overrightarrow{Q}, \overrightarrow{b_0}) &= \frac{\overrightarrow{Q} \odot \overrightarrow{b_0}}{|\overrightarrow{Q}| \cdot |\overrightarrow{b_0}|} = \frac{\sum_{i=1}^{n} q_i a_{0i}}{\sqrt{\sum_{i=1}^{n} q_i^2} \times \sqrt{\sum_{i=1}^{n} a_{0i}^2}} \\ cos(\overrightarrow{Q}, \overrightarrow{b_1}) &= \frac{\overrightarrow{Q} \odot \overrightarrow{b_1}}{|\overrightarrow{Q}| \cdot |\overrightarrow{b_1}|} = \frac{\sum_{i=1}^{n} q_i a_{1i}}{\sqrt{\sum_{i=1}^{n} q_i^2} \times \sqrt{\sum_{i=1}^{n} a_{1i}^2}} \end{aligned} \quad (1)$$

If $cos(\overrightarrow{Q}, \overrightarrow{b_0}) > cos(\overrightarrow{Q}, \overrightarrow{b_1})$, b_0 is more similar to Q; otherwise b_1 is more similar.

III.C. Popularity Aware Vector Space Models

III.C.1. Adding the Impact of Data Popularity

In order to give high priority to deliver more popular data, we assign values to the entries in the VSM matrix according to the popularity of the data. We denote the data set of a vehicle as $D = \{b_0, b_1, \cdots, b_m\}$ and the popularity score of data b_i as f_i ($f_i > 1$ and it is proportional to the popularity of b_i. The calculation of f_i will be discussed in Section III.C.2). Suppose the original VSM matrix, say $A_{m\times n}$, is as following:

$$A_{m\times n} = [a_{ij}]_{m\times n}, \text{ w.s.t. } 0 < i < m, 0 < j < n.$$

Then we get the entry in the Popularity Aware VSM (PVSM) matrix $a_{ij}^P = f_i \times a_{ij}$. So the new PVSM matrix can be computed as

$$A_{m\times n}^P = [f_i \times a_{ij}]_{m\times n}.$$

In PVSM, the length of the term vector of data item b_i is scaled by f_i according to its popularity. However, Equation 1 uses *cosine measure* to compute the similarity between the two term vectors, which normalizes

Table 1: An example of data tags.

Data ID	**Data tags**				
	File type	**Category**	**Other**	**Other**	**Other**
Data b_0	MP3	Music	John Lennon	Love	/
Data b_1	MP3	Music	John Lennon	Beetles	Yoko Ono
Data b_2	Video	Music	Pop	Mika	/

Table 2: VSM matrix generated for the example.

Data ID	**Terms**										
	MP3	**Video**	**Music**	**Pop**	**John Lennon**	**Mika**	**Yoko Ono**	**Beetles**	**Imagine**	**Love**	**... ...**
Data b_0	1	0	1	0	1	0	0	0	0	1	0 ... 0
Data b_1	1	0	1	0	1	0	1	1	0	0	0 ... 0
Data b_2	0	1	1	1	0	1	0	0	0	0	0 ... 0
Query Q	1	0	1	0	1	0	0	0	1	0	0 ... 0

both vectors to compare the angles of different vector pairs and discards the effect of the vector length. To add the impact of popularity, we revise Equation 1 and compute the relevance between the query vector $\overrightarrow{Q}$ and the data vector $\overrightarrow{b_i}$ with the production of their *cosine measure* and the popularity score of the data item b_i, i.e.,

$$\begin{aligned} relevance(Q, b_i) &= cos(\overrightarrow{Q}, \overrightarrow{b_i}) \times f_i \\ &= \frac{\sum_{i=1}^{n} q_i \times a_{ij} \times f_i}{\sqrt{\sum_{j=1}^{n} q_j^2} \times \sqrt{\sum_{j=1}^{n} a_{ij}^2}} \end{aligned} \quad (2)$$

Here, we also give another relevance function

$$relevance'(Q, b_i) = \frac{\sum_{i=1}^{n} q_i \times a_{ij} \times f_i}{\sqrt{\text{number of non-zero entries in } b_i}} \quad (3)$$

This function is much simpler and it needs less computation compared to the original relevance function. The following proof shows that $relevance'(Q, b_i)$ is equivalent to $relevance(Q, b_i)$.

Theorem 1. To compare the relevance between any data b_i and a given query Q, the relevance function $relevance'(Q, b_i)$ is equivalent to $relevance(Q, b_i)$.

Proof: For a given query Q, the first term of the denominator in $relevance(Q, b_i)$, $\sqrt{\sum_{j=1}^{n} q_j^2}$, is always a constant value to different data items. At the same time, in a binary matrix where the value of each entry is either 0 or 1, the second term of the denominator in $relevance(Q, b_i)$, $\sqrt{\sum_{j=1}^{n} a_{ij}^2}$, equals to the square root of the number of non-zero entries in the data vector of data item b_i. Then, it is obvious that

$$\begin{aligned} relevance(Q, b_i) &\propto \frac{\sum_{i=1}^{n} q_i \times a_{ij} \times f_i}{\sqrt{\sum_{j=1}^{n} a_{ij}^2}} \\ &= \frac{\sum_{i=1}^{n} q_i \times a_{ij} \times f_i}{\sqrt{\text{number of non-zero entry in } b_i}} \end{aligned}$$

To summarize, the relevance function $relevance'(Q, b_i)$ is equivalent to $relevance(Q, b_i)$ for the query-data relevance comparison. ■

Therefore, in Roadcast, we use the simplified $relevance'(Q, b_i)$ instead of $relevance(Q, b_i)$ as the relevance function for fast computation.

III.C.2. Calculating the Popularity Score f_i

f_i is used to represent the popularity of data item b_i. If one data item is more popular, its popularity score f_i should be larger. In our implementation, f_i is the estimated number of times that b_i is picked to reply queries during a given time period. The initial value of f_i is set to the number of times that the data is read by the local user during a given time period. Since f_i changes dynamically, we use a decay function that gives preference to more recent accesses and de-emphasizes the significance of past accesses in prediction. In particular, at the *(t+1)-th* time period, the popularity score of b_i is defined as

$$f_i(t+1) = \delta \times f_i(t) + (1 - \delta) \times F$$

where F is the number of times the data is accessed in the last time period and δ is the decay coefficient. In our experiments (see Section V), we set $\delta = 0.2$. f_i is recorded by individual vehicles in a distributed way. Thus, different vehicles may have different f_i for the same data item b_i.

III.C.3. Using Sparse Matrix Algorithm to Optimize Information Storage and Relevance Calculation

In VSM, the entry a_{ij} indicates the presence or absence of term i in data b_j. To precisely describe a

Table 3: The storage index structure for the example.

non_zero_vector	4	4	4	4	1	1	1	1	1	2	2	2	2
$column\_vector$	0	2	4	9	0	2	4	6	7	1	2	3	5
row_vector	0	4	9	13									

Table 4: Query processing and relevance ranking.

Data ID	**Relevance Score**	**Ranking**
b_0	$(4 \times 1 + 4 \times 1 + 4 \times 1 + 4 \times 0)/\sqrt{4} = 6$	1
b_1	$(1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 1 \times 0)/\sqrt{5} = 1.34$	2
b_2	$(2 \times 0 + 2 \times 1 + 2 \times 0 + 2 \times 0)/\sqrt{4} = 1$	3

query, the terms' pool can be a vocabulary dictionary in which the number of terms is quite large. Consequently, the size (the number of columns) of the PVSM matrix becomes huge, increasing the overhead for calculating the similarity and storing these data. Fortunately, we observe that PVSM is actually a sparse matrix where only a small number of keywords are used to represent both data and query while most others are absent in vector (i.e., most entries are 0). Hence, we can apply some techniques to optimize the data storage and relevance calculation.

Storage optimization: The sparse matrix stores only non-zero entries to save space. The index structure is stored in three sparse vectors. The first vector stores non-zero entries of the sparse matrix. The weight $(f_i \times a_{ij})$ in a particular data represents the importance of a term in a data. Hence we store all non-zero $(f_i \times a_{ij})$ for each element in the first non-zero vector. The second vector is the column vector, where each entry stores the term identifier or the column index for the corresponding term in non-zero vector. The third vector is the row vector that consists of pointers to each row of the matrix. The row vector consists of only one entry for each row of the matrix and the value of each entry is the position of the first non-zero entry of each row in non-zero vectors. For example, the storage index structure of Table 2 can be shown in Table 3, with data b_0, b_1, b_2 and query Q. We can see that it saves memory space because it stores only non-zero entry and only one entry for each row of the matrix.

Calculation optimization: The algorithm to optimize the computation overhead is shown in Algorithm 1. It starts by scanning the first value in vector $non\_zero\_vector[]$. This is the first non-zero term in the first data vector. If the corresponding term appears in the query vector $Query[]$, this value is added to the relevance function and it continues to scan the next non-zero value; otherwise it just continues the scan. When all non-zero terms of one data item are scanned,

Algorithm 1 : Relevance Calculation with the Vector Matrix Multiplication Algorithm

```
1: Input:
2:   M: Number of data in the memory;
3:   $non\_zero\_vector$[]: weight of each non-zero element;
4:   $row\_vector$[]: position of the first non-zero element of each
     row;
5:   $col\_vector$[]: column of each non-zero element in
     $non\_zero\_vector$;
6:   $Query$[]: query
7: Output:
8:   $Relevance$[]: the relevance value of each data to the query;
9:
10: FOR $(count = 0; count < M; count++)$
11:   $temp = 0;$
12:   FOR $(row = row\_vector[count];$
          $row <= (row\_vector[count+1] - 1);$
          $row++)$
13:     $col = col\_vector[row];$
14:     $temp = temp + non\_zero\_vector[row] \times Query[col];$
15:   END FOR
16:   $Relevance[count] = temp/sqrt(row\_vector[count+1]$
                                    $-row\_vector[count]);$
17: END FOR
```

the final relevance value of this data item can be calculated by dividing the current relevance value by the square root of the number of non-zero terms in the data vector. Clearly, the computation complexity of Algorithm 1 is $\mathcal{O}(n)$, where n is the number of elements in vector $non\_zero\_vector[]$.

The optimization can accelerate the query-data relevance calculation and save memory space. Using Algorithm 1 and the storage index of Table 3, the result of query processing and relevance ranking of the example are shown in Table 4. As can be seen, although both b_0 and b_1 match three keywords of query Q, b_0 has a higher relevance ranking due to its higher popularity factor and more concentrated terms.

IV. Popularity Aware Data Replacement

In the previous section, we proposed techniques to make popular data maintaining a high density in the network. At the same time, we need to make sure that popular data should not be replicated too aggressively and less popular data should not be totally removed from the network. [6] and [23] show that the *square-root* data allocation strategy can achieve optimal replication and minimize the query cost. In the square-root strategy, the number of data replications should be proportional to the square root of their popularity. In Roadcast, we propose a simple and cost-effective solution that can help achieve the square-root data allocation by using local data replacement. In this way, the popular data can have more, while not too many replications in the network and some less popular data can also be replicated to reduce the query delay. In this section, we first introduce a popularity aware data replacement algorithm and then prove that it can reach the optimal square-root data allocation.

IV.A. Data Allocation Principal

In an unstructured peer-to-peer system with blind search, we must answer the question: *how many copies of each data item should be in the system so that the search cost (in terms of query delay) for the data is minimized, assuming that the total amount of storage in the network is fixed?* This problem has been studied in [6] and [23]. We first review these results and illustrate the difficulty of achieving the strict optimal data allocation in a dynamic VANET system.

Consider the system model used in [6] and [23] where the network consists of n nodes (vehicles), each with capacity ρ which is the average number of data items that the node can hold. There are l available distinct data items and each item b_i is replicated at r_i random nodes. Suppose $R = \sum_{i=1}^{l} r_i$, where R is the total number of data copies in the network. Data b_i is requested with a rate p_i, where we normalize this by setting $\sum_{i=1}^{l} p_i = 1$, and obviously $p_i \propto f_i$. Query is delivered to any encountered node until the query can be served. Therefore, the number of encounter nodes required until the query is served is a *Geometric* random variable, and the probability $Pr(k)$ that the data is found on the k'th node follows the *Geometric distribution* G($\frac{r_i}{n}$) and it can be calculated as

$$Pr(k) = \frac{r_i}{n}(1 - \frac{r_i}{n})^{k-1}$$

Thus, the average search size A_i is the mean of G($\frac{r_i}{n}$), which is $\frac{n}{r_i}$. We are interested in the average search size of all available data items:

$$A = \sum_{i=1}^{n} p_i A_i = n \sum_{i=1}^{n} \frac{p_i}{r_i}$$

This metric essentially captures the query cost in terms of query delay in a VANET.

Uniform, Proportional and Square-Root Allocation: The simplest strategy is to create the same number of copies of each data item, i.e., $r_i = \frac{R}{l}$. This is the uniform allocation strategy. In this case the average search size $A_{uniform}$ is

$$\begin{aligned} A_{uniform} &= \sum_{i=1}^{n} p_i A_i \\ &= \sum_{i=1}^{n} p_i \frac{l}{\rho} \\ &= \frac{l}{\rho} \end{aligned}$$

which means for uniform allocation, the search size is independent of the query distribution.

In the proportional allocation, each data is replicated proportional to the access frequency, i.e., $r_i = Rq_i$. In this case the average search size is

$$\begin{aligned} A_{proportional} &= \sum_{i=1}^{n} p_i A_i \\ &= n \sum_{i=1}^{n} \frac{p_i}{R \cdot p_i} \\ &= \frac{l}{\rho} = A_{uniform} \end{aligned}$$

Based on the above results, we see that the Uniform and Proportional allocation strategies lead to the same search size, which means that these two strategies have the same query cost, and the query cost is independent of the query distribution.

The square-root allocation strategy assumes that the replica of each data in the network is proportional to the square-root of its access frequency, i.e., $r_i = \frac{R}{\sum_{i=1}^{n} \sqrt{p_i}} \cdot \sqrt{p_i}$. Then the average search size is

$$\begin{aligned} A_{squareroot} &= \sum_{i=1}^{n} p_i A_i \\ &= n \sum_{i=1}^{n} \frac{p_i}{\frac{R}{\sum_{i=1}^{n} p_i} \cdot \sqrt{p_i}} \\ &= n \sum_{i=1}^{n} \frac{\sqrt{p_i} \cdot \sum_{i=1}^{n} \sqrt{p_i}}{R} \\ &= \frac{1}{\rho} (\sum_{i=1}^{n} \sqrt{p_i})^2 \end{aligned}$$

Since $(\sum_{i=1}^{n} \sqrt{p_i})^2 \ll l$, $A_{squareroot}$ is considerably smaller than $A_{uniform}$ and $A_{proportional}$. Actually, [6] and [23] have revealed that square-root is the optimal data allocation strategy.

Achieving Square-Root Data Allocation in VANETs:

We use $E_i = \{e_{ij} | \text{for all } b_j\}$ to represent the state of vehicle v_i's buffer where:

$$e_{ij} = \begin{cases} 1 & \text{if } b_j \text{ is in the buffer of } v_i \\ 0 & \text{if } b_j \text{ is not found in } v_i \end{cases}$$

The access frequency of each vehicle to each data item is denoted as:

$$F = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1..l} \\ f_{21} & f_{22} & \cdots & f_{2..l} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nl} \end{pmatrix}$$

where $\sum_{j=1}^{l} f_{ij} = f_i \propto p_i$.

Furthermore, we use c_{ijk} to represent the cost in terms of query delay for vehicle v_i to access data b_j from vehicle v_k. Then, for vehicle v_i, its total access cost can be calculated as:

$$Cost_i = \sum_{j=1}^{d} f_{ij} \times min\{c_{ijk} | \text{for all } v_k\} \qquad (4)$$

Therefore, the goal of data allocation is to find the best replication arrangement in order to optimize the following objective function:

$$min\{\sum_{i=1}^{l} (Cost_i)\} \qquad (5)$$

subject to:

$$\sum_{j=1}^{l} e_{ij} \leq \rho \qquad \text{for all } v_i$$

and

$$\sum_{i=1}^{n} e_{ij} \propto \sqrt{p_i}$$

This allocation problem can be reduced to the multi-Knapsack problem (MKP) [7] that is known as NP-complete. Therefore, we present heuristics to provide square-root data allocation and near optimal performance only with local and distributed data replacement technique.

IV.B. The Popularity Aware Data Replacement Algorithm

In Roadcast each data item is stored locally after it has been downloaded to serve local requests. Each buffered data item is associated with a cost value. Intuitively, if one data item has more replications in the network, it will be easier to find and its access cost (delay) is low. When the memory is full, the data with the lowest cost value will be replaced by the newly obtained data. The idea of our data replacement comes from the GreedyDual-Size algorithm proposed by Cao and Irani in [5], which is used for web cache replacement. However, GreedyDual-Size could not capture and leverage the knowledge of the long-term access frequencies of different data. Recent studies have shown the prevalence of Zipf-like distributions in data access, which implies that the probability of future access depends on past access frequencies. Therefore, in the popularity aware data replacement algorithm, we incorporate the temporal popularity factor. Different from the web cache replacement algorithms, we use the latest retrieval delay to represent the access cost of one particular data. The proposed data replacement algorithm can help replace the most suitable data and achieve the global optimal data allocation in a distributed way.

The Algorithm:

We incorporate the temporal popularity factor (i.e., access frequency) into the original GreedyDual-Size algorithm through the use of a new cost value for each data. In Roadcast, the cost value H_i of data b_i is defined as the expected normalized cost saving as a result of having data b_i locally, i.e.,

$$H_i = \frac{c_i \times f_i}{s_i} \qquad (6)$$

where f_i is the popularity score (defined in Section III.C.2) of data b_i, c_i is its estimated retrieval cost (i.e., last retrieval delay), and s_i is the size of the data b_i.

A new value L, which equals to the lowest H value of all the data in local memory, is used as the "inflation" value in data replacement. When a new data item is brought in, its H value is set as its normalized access cost plus the L value. At the same time, if there is no memory space left, the data with the lowest H value has to be evicted and L is set to this H. Algorithm 2 presents the details of the data replacement algorithm.

Intuitively, if a data item has a higher retrieval delay due to its low replication density, based on this data replacement algorithm, it will be able to stay locally

Algorithm 2 : The Popularity-Aware Data Replacement Algorithm

```
1: Input:
2:   p: the data that is obtained;
3:   $f[]$: popularity score;
4:   $c[]$: retrieval cost (i.e., access delay of last retrieval);
5:   $s[]$: data size;
6:
7: INITIALIZE
8:   L=0.0;
9: FOR each obtained data $p$
10:   IF $p$ is in the memory
11:     $H_p = L + f_p \times c_p/s_p$;
12:   ELSE
13:     WHILE there is not enough free memory for $p$
14:       $L = min\{H_q \mid q$ is in the memory$\}$;
15:       Evict $q$ which satisfies $H(q) = L$;
16:     END WHILE
17:     Store $p$ in the memory;
18:     $H_p = L + f_p \times c_p/s_p$;
19:   END IF
20: END FOR
```

for a longer time. Meanwhile, a data item with high density in the network is more likely to be obtained from neighboring vehicles. Also its retrieval cost is low and its initial H_i will be small, which means it may be evicted easily. With this algorithm, the number of replications of different data items is controlled by the popularity factor.

Theorem 2. The popularity aware data replacement algorithm (Algorithm 2) can achieve the optimal square-root data allocation.

Proof: Assume the network consists of n vehicles, each with capacity ρ which is the number of data items that the vehicle can hold. Let r_i denote the number of replications of one particular data b_i. Then the density of data b_i, denoted as d_i, equals to $\frac{r_i}{n\times\rho}$. It is easy to see that d_i is a random variable evolving over time. When the replications of the data are evicted from the network, d_i decreases. When new copies are replicated in the system, d_i increases. It is not hard to see that when the local memory is all used by data replications,

$$\sum_{i=0}^{n-1} d_i = 1 \tag{7}$$

Then we have a dynamic system with a differential equation:

$$\frac{\mathbf{d}d_i}{\mathbf{d}t} = -\alpha d_i + \beta \times \frac{f_i}{d_i} \tag{8}$$

where α ($0 < \alpha < 1$) is the rate at which the copies of the data are evicted, and β is the density increasing constant. In Equation 8, $-\alpha d_i$ indicates that random copies are evicted and the density decreases linearly. $\beta\times\frac{f_i}{d_i}$ represents that each request for data b_i results in an increase of the density. The increase is proportional to both the access frequency f_i and its life time. In particular, the expected lifetime is proportional to the expected access cost (i.e., retrieval delay) in Equation 6. With the assumption that each vehicle queries its encountered vehicles to check if they have the interested content, the retrieval delay of one specific data b_i through such blind search is inversely proportional to the number of data replications (r_i) in the network, which is also proportional to the density of b_i (d_i), i.e.,

$$\text{life time of data item } b_i \propto c_i \propto \frac{1}{r_i} \propto \frac{1}{d_i} \tag{9}$$

By setting $\frac{\mathbf{d}d_i}{\mathbf{d}t} = 0$ in Equation 8, we can get the equilibrium point of this equation, i.e.,

$$\begin{aligned} &\frac{\mathbf{d}d_i}{\mathbf{d}t} = -\alpha d_i + \beta \times \frac{f_i}{d_i} = 0 \\ &\Rightarrow \ \alpha d_i = \beta \times \frac{f_i}{d_i} \\ &\Rightarrow \ \frac{\alpha}{\beta} \times d_i^2 = f_i \\ &\Rightarrow \ d_i \propto \sqrt{f_i} \end{aligned} \tag{10}$$

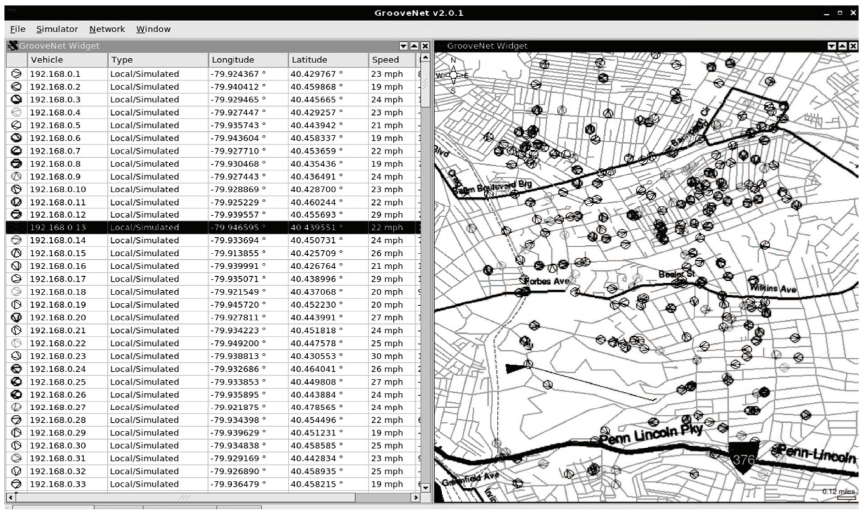
The result of Equation 10 shows the nonlinear system (Equation 8) converges to the square-root allocation at its steady state. Therefore, by using the proposed popularity aware data replacement algorithm, the data allocation obeys the optimal square-root rule. ■

V. Performance Evaluations

In this section, we evaluate the performance of the proposed Roadcast content sharing scheme and compare it to other solutions.

V.A. Simulation Setup

In our simulation setup, vehicles move within a fixed region of $3km \times 3km$. Each vehicle can initiate queries for some interested content. If the query cannot be served locally, it is sent to other encountered vehicles. When the requested data is sent back, the data is available to use. If the local memory of the vehicle is full, one or more data items will be evicted according to the data replacement algorithm. We implement Roadcast on the ns-2 simulator [2]. Since ns-2 is developed for generic ad hoc networks, it does not support VANET specific topologies and traffic control models. To provide a real VANET environment, we



	Vehicle	Type	Longitude	Latitude	Speed
	192.168.0.1	Local/Simulated	-79.924367 °	40.429767 °	23 mph
	192.168.0.2	Local/Simulated	-79.940412 °	40.459868 °	19 mph
	192.168.0.3	Local/Simulated	-79.929465 °	40.445665 °	24 mph
	192.168.0.4	Local/Simulated	-79.927447 °	40.429257 °	23 mph
	192.168.0.5	Local/Simulated	-79.935743 °	40.443942 °	21 mph
	192.168.0.6	Local/Simulated	-79.943604 °	40.458337 °	19 mph
	192.168.0.7	Local/Simulated	-79.927710 °	40.453659 °	22 mph
	192.168.0.8	Local/Simulated	-79.930468 °	40.435436 °	19 mph
	192.168.0.9	Local/Simulated	-79.927443 °	40.436491 °	24 mph
	192.168.0.10	Local/Simulated	-79.928869 °	40.428700 °	23 mph
	192.168.0.11	Local/Simulated	-79.925229 °	40.460244 °	22 mph
	192.168.0.12	Local/Simulated	-79.939557 °	40.455693 °	29 mph
	192.168.0.13	Local/Simulated	-79.946595 °	40.439551 °	22 mph
	192.168.0.14	Local/Simulated	-79.933694 °	40.450731 °	24 mph
	192.168.0.15	Local/Simulated	-79.913855 °	40.425709 °	26 mph
	192.168.0.16	Local/Simulated	-79.939991 °	40.426764 °	21 mph
	192.168.0.17	Local/Simulated	-79.935071 °	40.438996 °	29 mph
	192.168.0.18	Local/Simulated	-79.921549 °	40.437068 °	20 mph
	192.168.0.19	Local/Simulated	-79.945720 °	40.452230 °	20 mph
	192.168.0.20	Local/Simulated	-79.927811 °	40.443991 °	27 mph
	192.168.0.21	Local/Simulated	-79.934223 °	40.451818 °	24 mph
	192.168.0.22	Local/Simulated	-79.949200 °	40.447578 °	25 mph
	192.168.0.23	Local/Simulated	-79.938813 °	40.430553 °	30 mph
	192.168.0.24	Local/Simulated	-79.932686 °	40.464041 °	26 mph
	192.168.0.25	Local/Simulated	-79.933853 °	40.449808 °	27 mph
	192.168.0.26	Local/Simulated	-79.935895 °	40.443884 °	24 mph
	192.168.0.27	Local/Simulated	-79.921875 °	40.478565 °	24 mph
	192.168.0.28	Local/Simulated	-79.934398 °	40.454496 °	22 mph
	192.168.0.29	Local/Simulated	-79.939629 °	40.451231 °	19 mph
	192.168.0.30	Local/Simulated	-79.934838 °	40.458585 °	25 mph
	192.168.0.31	Local/Simulated	-79.929169 °	40.442834 °	23 mph
	192.168.0.32	Local/Simulated	-79.926890 °	40.458935 °	25 mph
	192.168.0.33	Local/Simulated	-79.936479 °	40.458215 °	19 mph

Figure 2: Simulation setup ($3km \times 3km$ area in Pittsburgh, PA)

use the GrooveNet simulator[1] [1] and a map of the *Pittsburgh* area (as obtained by the US Census Bureau data for street-level maps [3]) to generate the street topology (Figure 2) and vehicle mobility trace file. The mobility trace is used in the ns-2 simulations.

There are 150 or 300 moving vehicles following the street topology and the speed limits. 100 data items, with different data size (1, 3, or 5 units), are generated at the start of each simulation. Each vehicle can store up to 20 data units in its local memory but initially it randomly picks data items as its local data until the local memory is full. To describe the data content, the vocabulary dictionary consists of 40 different terms, and each data can randomly choose 2∼8 terms as its keywords. Similarly, each query consists of 3∼5 keywords from the same dictionary. The data access follows $Zipf$ distribution, where the access probability of the i^{th} term in the dictionary is represented as $P_i = \frac{1}{i^\theta \sum_{j=1}^{n} \frac{1}{j^\theta}}$, where $\theta \geq 0$, n is the dictionary size.

In Roadcast, the query requirement can be relaxed so that the data item that does not match all query requirements can still be used to serve the query. The default satisfaction degree is set to 75%, which means that if one query consists of 4 keywords, any data item that matches at least 3 of these 4 keywords can be used to serve the query. Most of the system parameters and their default values are listed in Table 5.

Roadcast consists of two components: the popularity aware content retrieval scheme and the popularity aware data replacement algorithm. To evaluate the performance of Roadcast, we compare it to three other content sharing schemes. The first two schemes use the same data replacement algorithm as Roadcast but different content retrieval schemes. Scheme I requires the data to be 100%-matched, while Scheme II relaxes the query requirement based on the satisfaction degree but without taking the popularity factor into consideration. Scheme III uses the same popularity aware content retrieval scheme as Roadcast but its data replacement is based on LRU (i.e., Least-Recently-Used). The performance of these content sharing schemes are measured by the query delay.

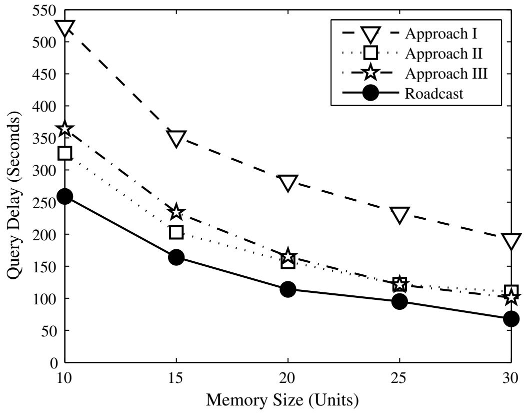
Table 5: Simulation configurations.

Parameter	**Default Value**
Simulation Time	20 minutes
Number of Vehicles	150, 300
Simulation Area	3km×3km (Pittsburgh)
Communication Range	200m
Data Size	1 unit, 3 units, 5 units
Memory Size	20 units
Keyword Set Size	40
Number of Keywords in Data Description	2∼8
Number of keywords in Query Description	3∼5
$Zipf$ Parameter θ	0.8
Satisfaction Degree	75%
Vehicle Speed	Street speed limit ±25%
Mobility Model	StreeSpeedModel [1]
Trip Model	SightSeeingModel [1]

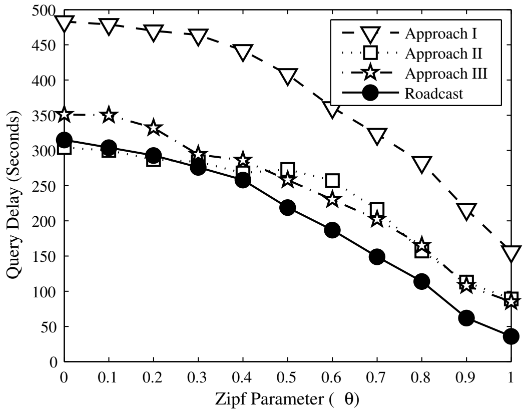
V.B. Query Delay

Figure 3 and Figure 4 compare Roadcast and other three content sharing schemes in terms of query delay in a 150-vehicle scenario and a 300-vehicle scenario, respectively.

[1]GrooveNet is a VANET simulator, which uses the map of the US Census Bureau's TIGER/Line 2000+ database [3] to generate a real city/street topology and provides a variety of useful models for mobility, traffic control, and etc, for VANET simulations. Therefore, we use GrooveNet to design the simulation scenario. We also rewrite the *logger* class of GrooveNet so that the logged mobility trace can be used in ns-2.

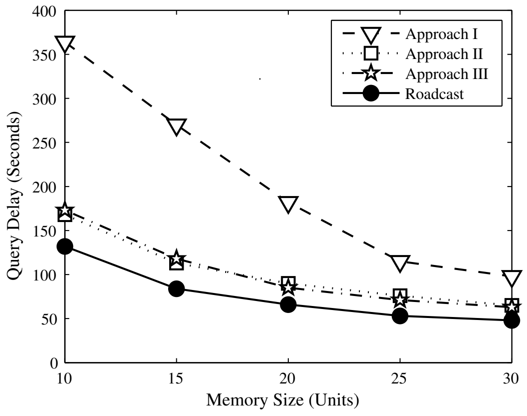


(a) Impact of memory size

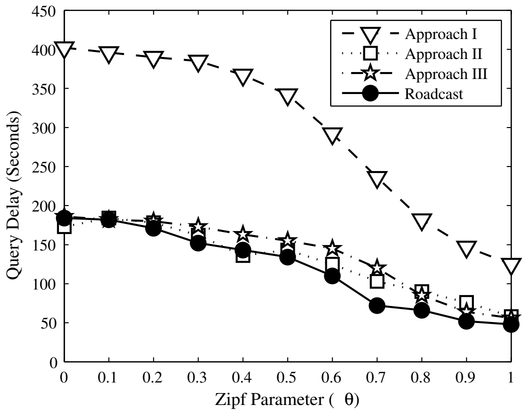


(b) Impact of content access skewness

Figure 3: Query delay in the 150-vehicle scenario



(a) Impact of memory size



(b) Impact of content access skewness

Figure 4: Query delay in the 300-vehicle scenario

As shown in Figure 3 (a), when the memory size is small (e.g., 10 units), all schemes have relatively higher query delay. When the memory size increases (e.g., to 30 units), the query delay decreases. This is because as the memory size increases, vehicles are able to buffer more data items. Hence, there will be more data replicas and the queries can be served by these replicas quickly. As shown in Figure 3 (a), Scheme I, which only accepts exactly matched data, has a much longer query delay than other three schemes (e.g., up to 175% of Scheme II, 190% of Scheme III, and 282% of Roadcast). This confirms the fact that it would take much longer time to find the exactly matched content in an intermittently connected VANET. Roadcast has the shortest query delay since it considers data popularity in content delivery and data replacement. It allows a more reasonable data distribution in the network, which further improves the data access performance. From the figure, we can see that Roadcast can save up to 32% and 38% query time compared to Scheme II and Scheme III, which either fails to consider popularity in content retrieval or ignores the popularity factor in data replacement.

From Figure 3 (a), we can also see that the query delay of Scheme II is much shorter than that of Scheme III when the memory size is small compared to that when the memory size is large. This is because when the memory size is small, the data replacement algorithm is much important. Moreover, LRU (used in Scheme III) does not consider the data size and its global popularity, but the popularity aware data replacement algorithm (used in Scheme II) achieves a better tradeoff between data size and popularity thus it can help achieve better performance. Figure 4 (a) shows similar results in the 300-vehicle scenario.

Figure 3 (b) compares the query delay of different schemes as a function of the content access skewness. In Zipf distribution, when θ=0, the access pattern is uniformly distributed, and different keywords have similar popularity. As θ increases, the access pattern becomes more skewed. As can be seen from the figure, when the content access is close to uniform distribution, the popularity aware content retrieval scheme and the data replacement algorithm do not have much

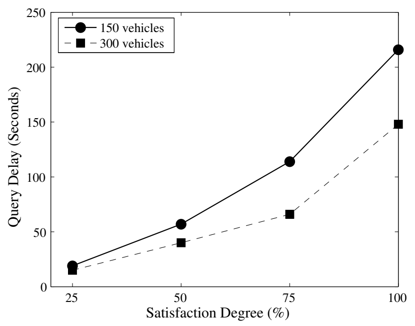
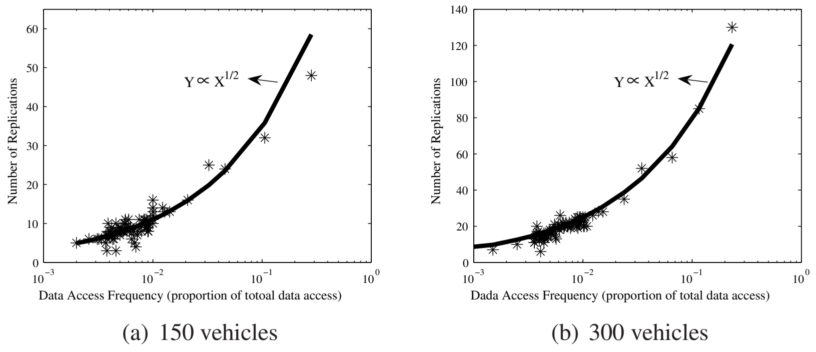


Figure 5: Impact of satisfaction degree



(a) 150 vehicles (b) 300 vehicles

Figure 6: Data allocation of Roadcast

advantage. But Scheme II, III and Roadcast still have much shorter query delay than Scheme I due to the relaxation on query requirement. As content access becomes skewed, Roadcast consistently outperforms other schemes. Here, the skewness of content access also helps data allocation. Therefore, as θ increases, the query delay decreases.

Meanwhile, the difference between Figure 3 and Figure 4 implies that vehicles can find the useful data more quickly in a dense VANET than in a sparse VANET.

V.C. Satisfaction Degree in Roadcast

In Roadcast, an important factor is to relax the query requirement so that users can have more choices to get the satisfying, but not exactly matched content quickly. Figure 5 illustrates how the satisfaction degree affects the performance. As the figure shows, when the satisfaction degree decreases, the query delay drops quickly. For example, when the satisfaction degree changes from 100%-match to 75%-match, the query delay can be reduced by 47% (150-vehicle) and 55% (300-vehicle). However, as the satisfaction degree decreases, the quality of the retrieved content may be degraded. Thus, there is a tradeoff between content quality and system performance.

V.D. Data Allocation

In Section IV, we prove that the popularity aware data replacement algorithm can achieve square-root data allocation in the system steady state. Here, we use simulation to verify it. Figure 6 plots the number of replicas for each data in the system at the end of the simulation, as a function of data access frequency. As can be seen from both Figures 6 (a) and (b), for those popular data items which have a high access frequency, they have more replicas than other less accessed data. Also the number of replicas for each data item closely follows the curve of the square-root function (the red curve). Consequently, the simulation results confirm that the popularity aware data replacement algorithm can help achieve the optimal square-root data allocation.

VI. Conclusions

This paper raises a simple question: how can we help users get the useful data as quickly as possible through vehicle-to-vehicle content sharing in an intermittently connected VANET? To answer this question, we propose Roadcast, a novel P2P content sharing scheme for VANETs. Roadcast relaxes the query requirement a little bit so that users can get the requested content quickly. Furthermore, Roadcast ensures more popular data is more likely to be shared with other vehicles. Roadcast consists of two components: popularity aware content retrieval and popularity aware data replacement. The popularity aware content retrieval scheme makes use of IR techniques to find the most relevant data towards user's query, but significantly different from IR techniques by taking the data popularity factor into consideration. To deal with the long delays of accessing the less popular data, we rely on the popularity aware data replacement algorithm, which can achieve the optimal *square-root* data allocation according to data popularity by only using local information.

This paper focuses on content sharing among independent vehicles. As future work, we are also interested in sharing content through cooperative retrieval and delivery among vehicles grouped as a "platoon" [9]. Besides, proactive caching is another important issue for content sharing in VANETs.

References

[1] Groovenet (hybrid-network simulator for vehicular networks). http://www.seas.upenn.edu

/ rahulm/research/groovenet/.

[2] Ns2 (the network simulator). http://www.isi.edu/nsnam/ns.

[3] U.S. Census Bureau. Tiger, tiger/line and tiger-related products, http://www.census.gov/geo/www/tiger/.

[4] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: routing for vehicle-based disruption-tolerant networks. *IEEE INFOCOM*, pages 1–11, April 2006.

[5] P. Cao and S. Irani. Cost-aware www proxy caching algorithms. *USENIX Symposium on Internet Technology and Systems*, pages 193–206, 1997.

[6] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts London, England, the second edition edition, 2001.

[8] S. Das, A. Nandan, G. Pau, M. Sanadidi, and M. Gerla. SPAWN: A Swarming Protocol For Vehicular Ad-Hoc Wireless Networks. In *ACM VANET*, 2004.

[9] D. Gerlough and M. Huber. Traffic flow theory - a monograph. *Special Report 165, Transporation Reseaerch Board*, 1975.

[10] S. Ghandeharizade, S. Kapadia, and B. Krishnamachari. Pavan: a policy framework for content availabilty in vehicular ad-hoc networks. In *ACM VANET*, pages 57–65, 2004.

[11] M. Guo, M. Ammar, and E. Zegura. V3: A vehicle-to-vehicle live video streaming architecture. In *IEEE PerCom*, 2005.

[12] S. Helal, N. Desai, and V. Verma. Konark: a service discovery and delivery protocol for adhoc networks. In *IEEE WCNC*, 2003.

[13] RDF Core Working Group http://www.w3.org/RDF/.

[14] Web Services Description Language (WSDL) Version 2.0 http://www.w3.org/TR/wsdl20.

[15] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *ACM SenSys*, 2006.

[16] J. Jeong, S. Guo, Y. Gu, T. He, and D. Du. Tbd: trajectory-based data forwarding for light-traffic vehicular networks. In *IEEE ICDCS*, pages 215–222, 2009.

[17] S. Jin and A. Bestavros. Popularity-aware greedydual-size web proxy caching algorithms. In *IEEE ICDCS*, 2000.

[18] M. Johnson, L. De Nardis, and K. Ramchandran. Collaborative content distribution for vehicular ad hoc networks. In *Allerton Conference Communication, Control, and Computing*, September 2006.

[19] D. Lee, H. Chuang, and K.Seamons. Document ranking and the vector-space model. *IEEE Software*, 14(2):67–75, 1997.

[20] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi. Dissemination and harvesting of urban data using vehicular sensing platforms. *IEEE Transactions on Vehicular Technology*, 2009.

[21] U. Lee, J. Park, J. Yeh, G. Pau, and M. Gerla. Code torrent: content distribution using network coding in VANET. In *ACM MobiShare*, 2006.

[22] U. Lee, J.-S. Park, E. Amir, and M. Gerla. Fleanet: a virtual market place on vehicular networks. *IEEE Transactions on Vehicular Technology*, September.

[23] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *International Conference on Supercomputing*, pages 84–95, 2002.

[24] A. Nandan, S. Dasand, G. Pau, M. Gerla, and M. Sanadidi. Co-operative downloading in vehicular ad-hoc wireless networks. In *IEEE/IFIP WONS*, pages 19–21, 2005.

[25] A. Nandan, S. Tewari, S. Das, M. Gerla, and L. Kleinrock. AdTorrent: delivering location cognizant advertisements to car networks. In *IEEE/IFIP WONS*, January 2006.

[26] P. Raghavan. Information retrieval algorithms: a survey. In *ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 11–18, 1997.

[27] H. Shen, Z. Li, T. Li, and Y. Zhu. Pird: P2p-based intelligent resource discovery in internet-based distributed systems. In *IEEE ICDCS*, pages 858–865, 2008.

[28] N. Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz. Routing in sparse vehicular ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, Oct. 2007.

[29] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter. Mddv: a mobility-centric data dissemination algorithm for vehicular networks. In *ACM VANET*, pages 47–56, 2004.

[30] W.H. Yuen, R.D. Yates, and S.-C. Mau. Exploiting data diversity and multiuser diversity in noncooperative mobile infostation networks. *IEEE INFOCOM*, pages 2218–2228, 2003.

[31] Y. Zhang, J. Zhao, and G. Cao. On scheduling vehicle-roadside data access. In *ACM VANET*, pages 9–18, 2007.

[32] Y. Zhang, J. Zhao, and G. Cao. Roadcast: a popularity aware content sharing scheme in vanets. In *IEEE ICDCS*, pages 223–230, 2009.

[33] J. Zhao and G. Cao. VADD: vehicle-assisted data delivery in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 57(3):1910–1922, May 2008.

[34] J. Zhao, Y. Zhang, and G. Cao. Data pouring and buffering on the road: a new data dissemination paradigm for vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 56(6):3266–3277, Nov. 2007.