

Roadcast: A Popularity Aware Content Sharing Scheme in VANETs

Yang Zhang, Jing Zhao and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
Email: {yangzhan,jizhao,gcao}@cse.psu.edu

Abstract

Content sharing through vehicle-to-vehicle communication can help people find their interested content on the road. In VANETs, due to limited contact duration and unreliable wireless connection, a vehicle can get the useful data only when it meets another vehicle and the encountered vehicle has the exactly matched data. However, the probability of such case 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 ensures popular data is more likely to be shared with other vehicles so that the overall query delay and the query hit ratio 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 and popular data towards user's query. The popularity aware data replacement algorithm ensures that the density of different data is proportional to their popularity in the system steady state, which firmly obeys the optimal "square-root" replication rule. Results based on real city map and real traffic model show that Roadcast outperforms other content sharing schemes in VANETs.

1. 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 [1]. With the support of peer-to-peer wireless communication, content can be shared among vehicles beyond the infrastructure coverage [2], [3]. 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 [2]–[4]. 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. Obviously, such chances are very low in VANETs. Although service discovery techniques [5] 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 [1], [6], 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 popular data which is densely disseminated in the network, it may take much shorter time than requesting rare data, because the chance of meeting one vehicle that has the popular 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 propose to disseminate data which can 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. But different from the traditional IR techniques, we consider the factor of data popularity 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 objects have to be replaced. The proposed data replacement algorithm ensures that the data replications with different popularity can have different life time so that popular data can have more, while not

too many, copies in the network. Our analysis shows that with the proposed data replacement algorithm, the densities of different data are proportional to their popularity in the system steady state, which firmly obeys the optimal “*square-root*” replication rule [7]. Simulation results show that the proposed popularity aware content sharing solutions can reduce the data access delay and improve the query hit ratio while satisfying the user requirement.

The rest of this paper is organized as follows. Section 2 describes the popularity aware content retrieval scheme. Section 3 presents the popularity aware data replacement algorithm used to achieve the optimal data allocation. Performance evaluations are shown in Section 4 and related work is discussed in Section 5. Finally, we conclude the paper in Section 6.

2. Popularity Aware Content Retrieval

2.1. Overview

In our popularity aware content retrieval, there are 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 to get less interested data (but still nice to have) with better chances or to get perfectly matching data with smaller chances and longer delay.

Considering the characteristics discussed above, we propose to give more opportunities to the content with higher popularity. For instance, 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 not only satisfy the neighboring vehicle’s request, but also 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 (1) *matching users’ query*, and (2) *increasing data accessibility in the future*. Different from other peer-to-peer content delivery

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	Ono
Data b_2	Video	Music	Pop	Mika	/

and sharing schemes, the decision on what data to deliver in Roadcast depends on both the client’s current interest and the overall demand in the network. A receiver is a content consumer who also carries the task of sharing the content with others. 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 objects based on keywords has been extensively studied in the IR community [8], [9]. However, their solutions are centralized and data accessibility is not an issue in their environments. In Roadcast, the basic idea is to leverage the Vector Space Model (VSM) to find the data that matches user’s query, and also consider data popularity. Thus, Roadcast may not always deliver the best matching data for a reply, instead it can deliver a less matching data if the data is more popular. Therefore more popular data is given more opportunities to be shared with others.

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

2.2. 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) or WSDL (i.e., Web Services Description Language) 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 content they want to retrieve, such as “MP3 music rock John Lennon Imagine”. Similarly, each data object 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 / John Lennon / Imagine”. The tag of a data item 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 user’s query. Suppose there are m data objects in a vehicle, and each data object 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 object b_i can be represented by a binary vector in the n -dimensional space, say \vec{b}_i , whose entry indicates the presence or absence of one particular term in the tags of data object b_i . The entry is “0” if the term does not occur in the data tag, and “1” otherwise.

Table 2. The VSM Matrix Generated For The Example

Data ID	Terms										
	MP3	Video	Music	Pop	John Lennon	Mika	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

In this way, all data in a vehicle can be represented by a $m \times n$ binary matrix, where every row represents one data object (shown in Table 2). Similarly, a query can also be thought as a vector in the same space. Thus, content retrieval becomes a matter of finding the data vectors in the space that are closest to the query vector.

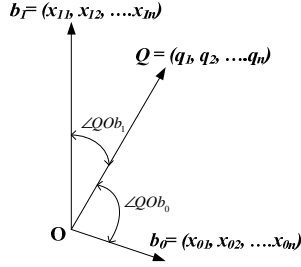


Figure 1. Similarity of Vectors

To answer a query, the data objects are ranked according to the similarity between the data vector and the query vector. A common measure of the similarity between two binary vectors with the same dimension is to calculate the number of their overlapped “1”s. However, this method has bias since the data objects with more terms tend to be ranked higher than those with fewer terms. We use the angle of two vectors to represent their similarity, which is able to remove the bias due to the number of terms. For easy calculation, the angle of two vectors can be transformed to the cosine value of the angle. Formally, as shown in Figure 1, given a n -dimensional term vector of a query Q , $\vec{Q} = (q_1, q_2, \dots, q_n)$ and a data object, $\vec{b}_i = (x_{i1}, x_{i2}, \dots, x_{in})$, the similarity between query Q and the data objects b_i can be defined in Equation 1.

$$\cos(\vec{Q}, \vec{b}_i) = \frac{\vec{Q} \cdot \vec{b}_i}{|\vec{Q}| \cdot |\vec{b}_i|} = \frac{\sum_{j=1}^n q_j x_{ij}}{\sqrt{\sum_{j=1}^n q_j^2} \times \sqrt{\sum_{j=1}^n x_{ij}^2}} \quad (1)$$

If $\cos(\vec{Q}, \vec{b}_0) > \cos(\vec{Q}, \vec{b}_1)$, b_0 is more similar to Q ; otherwise b_1 is more similar.

2.3. Popularity Aware Vector Space Model

2.3.1. 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 data popularity. We denote the data set of a vehicle as $D = \{b_0, b_1, \dots, b_m\}$ 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 2.3.2). Suppose the original VSM matrix, say $A_{m \times n}$, is as following:

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

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 object 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 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 \vec{Q} and the data vector \vec{b}_i with the production of their *cosine measure* and the popularity score, i.e.,

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

Here, we also have another relevance function

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

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

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

Proof: For a given query Q , the first term of the denominator in $\text{relevance}(Q, b_i)$, $\sqrt{\sum_{j=1}^n q_j^2}$, is always a constant value to different data objects. 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 $\text{relevance}(Q, b_i)$, $\sqrt{\sum_{j=1}^n a_{ij}^2}$, equals to the square root of the number of non-zero entries in the data vector of data object b_i . Then,

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

To summarize, $\text{relevance}'(Q, b_i)$ and $\text{relevance}(Q, b_i)$ are equivalent for the relevance comparison. ■

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

Table 3. The Storage Index Structure

<i>non_zero</i>	4	4	4	4	1	1	1	1	1	2	2	2	2
<i>column</i>	0	2	4	9	0	2	4	6	7	1	2	3	5
<i>row</i>	0	4	9	13									

2.3.2. Calculating the Popularity Score f_i .

f_i is used to represent the popularity of data object b_i . If one data object 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 that the data is accessed in the last time period and δ is the decay coefficient. In our experiments (see Section 4), 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 object b_i .

2.3.3. 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 object b_j . To precisely describe a 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. We observe that PVSM is actually a sparse matrix where only a small number of keywords are used to represent data and query while most others are absent in vector (i.e., 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 the non-zero vector. For example, the storage index structure of Table 2 can be shown in Table 3, with data objects b_0 , b_1 , b_2 and query Q . 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

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 non-zero element in non_zero_vector;
6: Query[:]: query
7: Output:
8: Relevance[:]: the relevance value of each data object 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] × Query[col];
15:   END FOR
16:   Relevance[count] = temp/sqrt(row_vector[count + 1] -
                                row_vector[count]);
17: END FOR

```

Table 4. Query Processing and Relevance Ranking

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

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 object are scanned, the final relevance value of this data object 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 is $\mathcal{O}(n)$, where n is the number of elements in *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 (Table 2) are shown in Table 4. As can be seen, although data b_0 and b_1 both match three keywords of query Q , data b_0 has a higher relevance ranking due to its higher popularity factor and more concentrated terms.

3. Popularity Aware Data Replacement

In the previous section, we proposed techniques to have popular data maintain a high density in the network. At the same time, we need to make sure that popular data should not be replicated too aggressively. Cohen and Shenker [7] show that the *square-root* data allocation strategy, where the number of data replications is proportional to the square root of their popularity, has the optimal replication performance in minimizing the query cost. In Roadcast, we propose a simple and cost-effective solution that can help achieve the square-root data allocation by using local data replacement.

3.1. Overview

In Roadcast each data object is stored locally after it has been downloaded to serve local requests. Each buffered data is associated with a cost value, which is proportional to the time delay to get the data. Intuitively, if a data object 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 [10], 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. The proposed data replacement algorithm can help replace the most suitable data and achieve the global optimal data allocation in a distributed way.

3.2. 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 object b_i is defined as the expected normalized cost saving as a result of having data b_i locally, i.e.,

$$H_i = \frac{c_i \times f_i}{s_i} \quad (4)$$

where f_i is the popularity score (defined in Section 2.3.2) of data object 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 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 has to be evicted and L is set to this H .

Based on this algorithm, intuitively, if a data object has a higher retrieval delay due to its low replication density, based on this data replacement algorithm, it will be able to stay local for a longer time. Meanwhile, a data object with high density in the network is more likely to be obtained from neighboring vehicles. 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 objects is controlled by the popularity factor.

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

Proof: Assume the network consists of n vehicles, each with average capacity ρ which is the number of data objects that the vehicle can hold. Let r_i denote the number of replications of one particular data object 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 occupied by data replications, $\sum_{i=0}^{n-1} d_i = 1$. Then we have a dynamic system with a differential equation:

$$\frac{dd_i}{dt} = -\alpha d_i + \beta \times \frac{f_i}{d_i} \quad (5)$$

where α ($0 < \alpha < 1$) is the rate at which the data copies are evicted, and β is the density increasing constant. In Equation 5, $-\alpha d_i$, indicates random copies are evicted and the density decreases linearly, and $\beta \times \frac{f_i}{d_i}$ represents 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 4. With the assumption that each vehicle queries its encountered vehicles to check if they have its interested content, the retrieval delay of one specific data object 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.,

$$\text{life time of data object } b_i \propto c_i \propto \frac{1}{r_i} \propto \frac{1}{d_i} \quad (6)$$

By setting $\frac{dd_i}{dt} = 0$ in Equation 5, we can get the equilibrium point of this equation, i.e.,

$$\frac{dd_i}{dt} = -\alpha d_i + \beta \times \frac{q_i}{d_i} = 0 \Rightarrow d_i \propto \sqrt{f_i} \quad (7)$$

The result of Equation 7 shows that the nonlinear system (Equation 5) 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. ■

4. Performance Evaluations

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

4.1. Simulation Setup

In our simulation setup, vehicles move within a fixed region of $3km \times 3km$. Each vehicle can initiate queries for its 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 objects will be evicted according to the data replacement algorithm. We

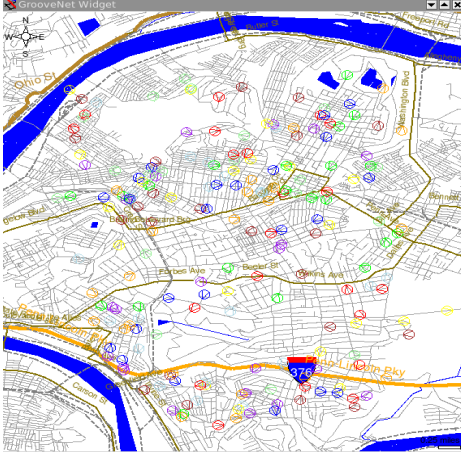


Figure 2. Simulation Setup

implement Roadcast on the ns-2 simulator [11]. 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 use the GrooveNet simulator [12] and a map of the *Pittsburgh* area (as obtained by the US Census Bureau data for street-level maps [13]) to generate the street topology (Figure 2) and vehicle mobility trace file. The mobility trace is used in the ns-2 simulations.

There are totally 150 moving vehicles following the street topology and the speed limits. 100 data objects, 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 objects 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.

In Roadcast, the query requirement can be relaxed so that the data object 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 if one query consists of 4 keywords, any data object 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 must 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).

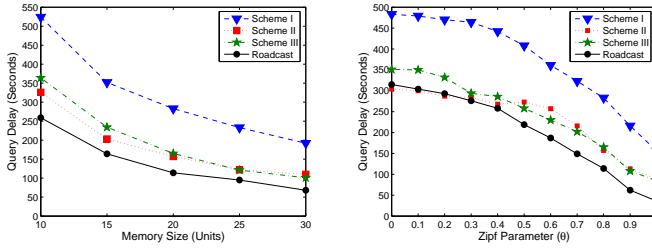
Table 5. Simulation Configurations

Parameter	Default Value
Simulation Time	20 minutes
Number of Vehicles	150
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
<i>Zipf</i> Parameter θ	0.8
Satisfaction Degree	75%
Vehicle Speed	Street speed limit $\pm 25\%$
Mobility Model	StreetSpeedModel [12]
Trip Model	SightSeeingModel [12]

By comparing Roadcast to Scheme I and Scheme II, we can see the advantage of the popularity aware content retrieval scheme. Scheme III can be used to demonstrate the benefit of the proposed data replacement algorithm. The performance of these content sharing schemes are measured by the query delay and the query hit ratio.

4.2. Query Delay

The query delay is defined as the average delay from initiating the query to receiving the satisfactory data. Figure 3 compares Roadcast and other three content sharing schemes in terms of query delay. As shown in Figure 3(a), when the memory size is small (e.g., 10 units), all schemes have a 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 objects. 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 when the memory size is large. This is because when the memory size is small, the data replacement algorithm is much important compared to that when the memory size is large. Further, 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



(a) Impact of memory size (b) Impact of access skewness

Figure 3. Query Delay

popularity thus it can help obtain a better performance.

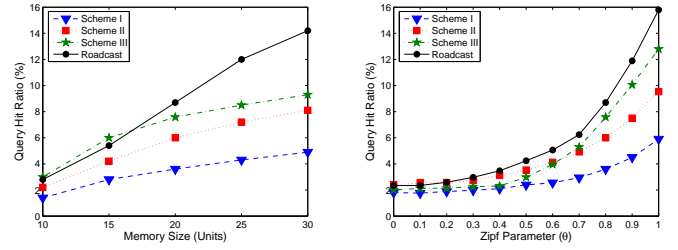
Figure 3(b) compares the query delay of different schemes as a function of the content access skewness. In Zipf distribution, when $\theta=0$, the access pattern is uniformly distributed, and different keywords have similar popularity. As θ increases, the access pattern becomes more skewed. As can be 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 advantage. But Scheme II, III and Roadcast still have much shorter query delay than Scheme I due to their relaxation on the query requirement. As content access becomes skewed, Roadcast consistently outperforms other schemes. Here, the skewness of content access also helps data allocation. As θ increases, the query delay decreases.

4.3. Query Hit Ratio

The query hit ratio is the possibility that the query can be served locally. Figure 4 evaluates four schemes on the query hit ratio as a function of memory size and data skewness. Again, Roadcast has a much higher query hit ratio than other three schemes, up to 190%, 61% and 53% higher than Scheme I, II, and III, respectively. This advantage of Roadcast may be due to two reasons: (1) during content retrieval, vehicles favor more popular data which increases the number of copies of the popular data objects in the network, and (2) when the memory is full, the data object with the smallest cost value (i.e., H_i , defined by Equation 4) will be evicted first, which considers both data popularity and data size factors. Therefore, for the skewness of content access, more query requests can be served locally in Roadcast, increasing its query hit ratio. We also observe that Scheme III outperforms Scheme II in query hit ratio. This means that the popularity aware content retrieval has more benefits on query hit ratio than data replacement in our simulation. Similarly, from Figure 4(b) we can see that as content access becomes skew, Roadcast outperforms other schemes. All these advantages come from the combination of popularity aware content retrieval and data replacement.

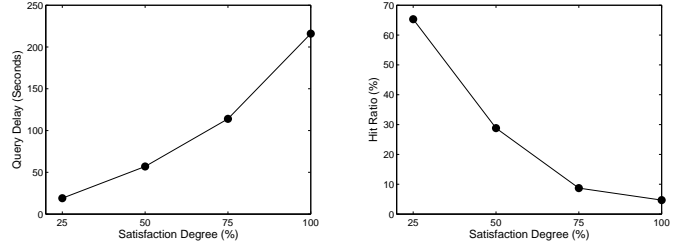
4.4. Satisfaction Degree in Roadcast

In Roadcast, an important factor is to relax the query requirement so that users can have more choices to get



(a) Impact of memory size (b) Impact of access skewness

Figure 4. Query Hit Ratio



(a) Query Delay (b) Query Hit Ratio

Figure 5. Impact of Satisfaction Degree

satisfactory but not exactly matched content quickly. Figure 5 illustrates how the setting of satisfaction degree affects performance. As the figure shows, when the satisfaction degree decreases (i.e., the users will be easier to be satisfied), the query delay drops and the query hit ratio increases quickly. For example, when the satisfaction degree changes from 100%-match to 75%-match, the query delay can be reduced by 47% and the query hit ratio can rise from 4.7% to 8.6%. However, as the satisfaction degree decreases, the quality of the retrieved content may be degraded. There is always a tradeoff between the content quality and performance. The satisfaction degree should not be too low in real applications.

4.5. Data Allocation

In Section 3, we prove that the popularity aware data replacement algorithm can achieve the square-root data allocation in the system steady state. Here, we use simulations to verify it. Figure 6 plots the number of replications for each data object in the system at the end of the simulation, as a function of data access frequency. As we can see, for those popular data objects with a high access frequency, they have more data replications in the network than other less accessed data. Also the number of replications for each data closely follows the curve of a 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.

5. Related Work

Vehicular Networks: Vehicular networks represent an interesting application scenario not only for traffic safety and efficiency but also for more commercial and entertainment support [14], [15]. So far, however, most vehicular network

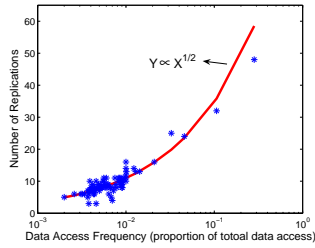


Figure 6. Data Allocation of Roadcast

researches focus on routing issues [1], [16], [17]. They all assume the consumer is known beforehand so that the sender can route the content to its destination. For example, VADD [1] studies how to choose the best routing path based on the traffic information. Zhao *et al.* [4] introduce data pouring and buffering techniques to disseminate data along the roads. This paper studies content sharing, where each vehicle queries useful data from its encountered neighbors. Different from destination aware routing and dissemination, how to retrieve the most suitable data from neighboring vehicles is the main focus of this paper.

Content Retrieval: In the last couple of years there has been an increasing interest in content retrieval through intermittent contact opportunities in vehicular networks. Guo *et al.* [2] study content retrieval in a small area, where vehicles in adjacent lanes exchange information as they pass through one another. Zhang *et al.* [15] analyze the scheduling issues of content retrieval at the road intersection. Our work, however, investigates the problem on the perspective of the whole network. Lee *et al.* [3] propose to accelerate content retrieval using randomized network coding. However, the network coding based data diffusion brings in large amount of redundant data which may not be useful but taking much communication bandwidth and memory space. In Roadcast, content retrieval is based on user's query request. We study how to efficiently share content with future encountered vehicles based on local information.

Data Replacement: Studies in data replacement start from cache replacement. Cao and Irani [10] study several replacement algorithms for web cache. They propose the GreedyDual-Size algorithm that enables the cache replacement to be sensitive to both the variability in data size and the retrieval cost. Later, Jin *et al.* [18] improve GreedyDual-Size by adding the popularity factor. However, all these works are based on web cache which is in a centralized environment. Roadcast differs from the existing works in that it is a distributed replacement algorithm and it aims to optimize the network-wide content sharing performance.

6. Conclusions

This paper raises a simple question: how can we help a user 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 user's 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. First, 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 data popularity into consideration. Second, the popularity aware data replacement algorithm is proposed to achieve the optimal *square-root* data allocation according to data popularity by only using local information. Simulation results show that Roadcast can reduce the data access delay and improve the query hit ratio while satisfying the user requirement.

References

- [1] J. Zhao and G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," *Proceedings of IEEE INFOCOM'06*, pp. 1–12.
- [2] M. Guo, M. Ammar, and E. Zegura, "V3: A vehicle-to-vehicle live video streaming architecture," *Proceedings of PerCom'05*, pp. 171–180, 2005.
- [3] U. Lee, J. Park, J. Yeh, G. Pau, and M. Gerla, "Code torrent: content distribution using network coding in VANET," *Proceedings of MobiShare'06*, pp. 1–5, 2006.
- [4] J. Zhao, Y. Zhang, and G. Cao, "Data pouring and buffering on the road: A new data dissemination paradigm for vehicular ad hoc networks," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 6, pp. 3266–3277, Nov. 2007.
- [5] H. Shen, Z. Li, T. Li, and Y. Zhu, "Pird: P2p-based intelligent resource discovery in internet-based distributed systems," *Proceedings of ICDCS'08*, pp. 858–865, 2008.
- [6] N. Wisitpongphan, F. Bai, *et al.*, "Routing in sparse vehicular ad hoc wireless networks," *IEEE JSAC*, Oct. 2007.
- [7] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 177–190, 2002.
- [8] D. Lee, H. Chuang, and K. Seamans, "Document Ranking and the Vector-Space Model," *IEEE Software*, vol. 14, no. 2, 1997.
- [9] P. Raghavan, "Information retrieval algorithms: a survey," *Proceedings of ACM-SIAM SODA'97*, 1997.
- [10] P. Cao and S. Irani, "Cost-aware www proxy caching algorithms," *Proceedings of the USENIX'07*, pp. 193–206, 1997.
- [11] "The network simulator. <http://www.isi.edu/nsnam/ns>."
- [12] "Groovenet hybrid-network simulator for vehicular networks. <http://www.seas.upenn.edu/rahulm/research/groovenet/>."
- [13] U. C. Bureau, "Tiger, tiger/line and tiger-related products."
- [14] S. Ghandeharizade, S. Kapadia, and *et al.*, "Pavan: a policy framework for content availability in vehicular ad-hoc networks," *Proceedings of VANET'04*, pp. 57–65, 2004.
- [15] Y. Zhang, J. Zhao, and G. Cao, "On scheduling vehicle-roadside data access," *Proceedings of VANET'07*, 2007.
- [16] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Max-prop: Routing for vehicle-based disruption-tolerant networks," *Proceedings of INFOCOM'06*, pp. 1–11, April 2006.
- [17] H. Wu, R. Fujimoto, *et al.*, "Mddv: a mobility-centric data dissemination algorithm for vehicular networks," *Proceedings of VANET'04*, 2004.
- [18] S. Jin and A. Bestavros, "Popularity-aware greedydual-size web proxy caching algorithms," *Proceedings of ICDCS*, 2000.