Distributed Storage Codes Reduce Latency in Vehicular Networks

Maheswaran Sathiamoorthy, *Student Member, IEEE,* Alexandros G. Dimakis, *Member, IEEE,* Bhaskar Krishnamachari, *Member, IEEE,* and Fan Bai *Member, IEEE,*

Abstract—We investigate the benefits of distributed storage using erasure codes for file sharing in vehicular networks through both analysis and realistic trace-based simulations. We show that the key parameter affecting the on-demand file download latency is the ratio of file size to download bandwidth. When this ratio is small so that a file can be communicated in a single encounter, we find that coding techniques offer very little benefit over simple file replication. However, we analytically show that for large ratios, for a memoryless contact model, distributed erasure coding yields a latency benefit of N/α over uncoded replication, where N is the number of vehicles and α the redundancy factor. Effectively, in this regime, coding yields the same performance as replicating all the files at all other vehicles, but using much less storage. We also evaluate the benefits of coded storage using large real vehicle traces of taxis in Beijing and buses in Chicago. These simulations, which include a realistic radio link quality model for a IEEE 802.11p dedicated short range communication (DSRC) radio, validate the observations from the analysis, demonstrating that coded storage dramatically speeds up the download of large files in vehicular networks.

Index Terms—Vehicular networks, erasure coding.

1 Introduction

The recent development of the IEEE 802.11p WAVE (Wireless Access in Vehicular Environment) protocol [1] and the allocation of Dedicated Short Range Communications (DSRC) spectrum, have increased interest in vehicular networking. In the United States, the FCC has allocated 75MHz of spectrum in the 5.9GHz band exclusively for vehicular networks and in Europe, the ETSI has allocated a 20MHz range in the same band. These bands enable vehicle-to-vehicle communication as well as vehicle-to-infrastructure (and vice versa) communication, and capabilities like these open up a number of possibilities. Most applications focus on safety, such as avoiding rear-end collisions; extended braking [1], [2]; and detecting and disseminating information about potholes, bumps and other anomalous road conditions [3]. Recently, applications that concern entertainment and file sharing are also receiving attention and involve different challenges (e.g., AdTorrent [4], CarTorrent [5], FleaNet [6], C2P2 [7]).

Content access and vehicle file sharing would enable users to access movies, music, videos, and other relevant content. In this paper we investigate the possibility of exploiting inter-vehicular communication to enable P2P file sharing without the use of access points (APs). One possibility for content access is to use the cellular infrastructure, but recent reports suggest that with the increasing use of smart phones, cellular data bandwidth is likely to remain limited and expensive [8]. Another option is to use APs, but they may be hard to deploy in high densities. In addition, due to the latency of content access from the Internet, a vehicle quickly passing by an AP might not have sufficient time to download its desired data. In contrast, the WAVE/WAVE BSS modes of IEEE 802.11p allow for rapid vehicle-to-vehicle file transfers [1] over potentially longer contact durations (e.g., if the vehicles are traveling in the same direction). Nevertheless, we note that the coded storage techniques we explore and analyze in this paper could also be used in a heterogeneous network architecture which integrates vehicle-to-vehicle communication with vehicle-to-infrastructure communication.

We identify two basic dissemination schemes that can be employed for data transfers in vehicular networks - one is the well studied ([4], [5], [9]–[13]) *push-based* mechanism and the other is a *pull-based* retrieval scheme which we study in this paper. As the names indicate, in push-based schemes, content is pushed into the networks, whereas in pull-based schemes, content is pulled from the network. It may be noted that push-based mechanisms work well for small sized data transfers such as traffic updates, pothole monitoring and other content that might be of interest to all users, but would fail to perform well for large file transfers of interest to only a few users (e.g., movies, long videos). This can be seen easily because traditional push-based schemes involve replicating the same file over multiple relays (e.g., epidemic routing [9], spray and wait [10], file swarming [4], [5]), which can be quite inefficient for large files. Other methods have been proposed such as the use of erasure coding [11], [12] and network coding [13], and even though they help reduce the delay and enhance the reliability, they fundamentally work by pushing more data than the file size into the network.

• *Maheswaran Sathiamoorthy, Alexandros G. Dimakis and Bhaskar Krishnamachari are with the department of Electrical Engineering, University of Southern California, Los Angles, CA 90089.*
E-mail: msathiam@usc.edu, dimakis@usc.edu, bkrishna@usc.edu
• *Fan Bai is with General Motors R&D, Warren, MI.*
E-mail: fan.bai@gm.com

This research was sponsored in part by General Motors Research and Development; the U.S. Army Research Laboratory under the Network Science Collaborative Technology Alliance, Agreement Number W911NF-09-2-0053; NSF Career Grant 1055099; a Research Gift by Microsoft Research and NSF Award 1217260.

The application considered here is close in spirit to a vehicular network based movie-rental application, similar to Netflix for mail. The application is not real-time, similar to how it may take a few days before a Netflix movie arrives in the mail. For such an application, there is no strict latency requirement, however, it is still desirable to minimize the average latency. The movies are stored in set of vehicles, such as taxis or buses in a city, to which the rental company has access. We call these as the seed vehicles. Other vehicles, which may have subscribed to the movie-rental application will be able to opportunistically download movies from these seed vehicles as they drive around. We call these as the sinks.

One can consider the entire storage of all the seed vehicles as a distributed repository. Given a set of files, we try to answer in this work the specific question of how to store the files in the repository.

Hence in this work, we consider a P2P file sharing application where the files are stored in the seed nodes as a distributed repository and sinks retrieve these files on-demand. The amount of data downloaded by a sink is no more than the file size (excluding control data), and thus such a pull based scheme is not only efficient but will also scale well with the number of nodes, file size etc. But in order to improve the latency and reliability of content access, intuitively the files should be stored with some redundancy. Thus, in order to reduce the latency of file access, we shift the burden from the expensive bandwidth to the relatively inexpensive storage, thereby enabling additional applications to run. Previously Kapadia *et al.* [7] have suggested a similar scheme, where the content is stored using simple uncoded replication. The novel contribution of our work is that we recommend the use of erasure codes, especially for large files, and we quantify the performance improvement of coded storage compared to uncoded replication.

We consider out of scope of this paper the orthogonal problem of the process by which the file repository is initially created and maintained. For this purpose other previously proposed schemes such as coded dissemination [14] or direct infrastructure download could be used. When the inter-vehicle communication data rates are high, or when the communicated files are sufficiently small, we find that simply storing multiple copies of each file has almost identical performance to an optimized erasure coded representation. However, we show that in other cases, when the file sizes are large compared to the download bandwidth, a distributed coded representation offers very substantial benefits and decreases the average download time by orders of magnitude.

Our analytical contribution is a novel probabilistic analysis of the latency for replicated and encoded distributed storage for vehicular networks. We analyze the expected delay for a vehicle trying to collect pieces to reconstruct a desired file by meeting other vehicles according to a memoryless process. We show how both replicated and encoded storage correspond to different balls and bins processes and using stochastic dominance and coupling arguments on these processes, we bound the expected download time. We identify three regions of interest depending on how the communication bandwidth per vehicle interaction d, compares to the file size $\mathcal{M}$ and the vehicle storage capacity per file $\mathcal{C}/m$ (here $\mathcal{C}$ is the storage per node and m is the number of files, and the setup is described in detail below). Our most surprising result is in the *bandwidth limited regime*: when $d < \mathcal{C}/m$. For this case we show that distributed erasure coding yields a latency of $\mathcal{M}/d$, which is equivalent to replicating all the files in all the vehicles. While coding uses much less storage, the equivalent set up of uncoded replication performs N/α times worse, where N is the number of vehicles and α is redundancy factor.

Beyond our analytical model, we present a comprehensive performance analysis using a real vehicle trace consisting of 1,000 taxis in Beijing and 1,608 buses in Chicago, combined with a realistic 802.11p DSRC Packet Delivery Ratio (PDR) model. These simulations validate the key insights from the analysis, demonstrating that coded storage substantially improves the timeliness of file downloads particularly in the bandwidth limited regime for large files. For instance, when using the Beijing dataset, we show that for downloading 1GB files, by the time 80% of the nodes are able to completely download the file under coded storage, only 4.4% of the nodes succeed if uncoded replication is used.

2 Background and Related Work

The fundamental goal of this paper is to analyze and minimize the delay in downloading files in a pull-based P2P vehicular content storage system. In contrast to prior work such as [7] which assumes that the content in such a system is stored using uncoded replication, we advocate and analyze the performance when the content is stored using erasure codes. Erasure coding consists of separating each file into k chunks and from these generating $n > k$ chunks of the same $\mathcal{M}/k$ size. If a Maximum Distance Separable (MDS) erasure code [15] is used, *any k out of the n encoded chunks suffice to reconstruct the original file*. One specific family of codes that are almost-MDS and are suitable for our application are digital fountain codes. Initially proposed by Byers *et al.* [16] and later developed by Luby [17] and Shokrollahi [18], digital fountain codes are binary near-MDS (almost all sets of $k(1+\epsilon)$ chunks suffice to reconstruct the file with high probability, but we neglect ϵ for simplicity) and have very fast and simple encoding and decoding algorithms. Using ideas related to this paper, fountain code designs were introduced for sensor network problems by Dimakis *et al.* [19] and Kamra *et al.* [20].

Erasure coding and uncoded replication have previously been compared in other contexts. For instance, [21] compares coding and uncoded replication for distributed storage in a wired system, and argues that coding is a clear winner as it provides mean times to failure that are magnitudes higher than that provided by replication. Another work [22] analyzes and compares these two approaches from the perspective of their ability to provide content availability in a P2P distributed hash table. They indicate that, given its complexity, erasure coding is useful only when the servers are extremely unreliable. Our work on vehicular networks has a different focus, not on ensuring availability in the face of server failures, but rather on reducing latency in the face of sparse and short-duration vehicular encounters. In this setting, we argue that coding is indispensable, particularly for large files.

Because they involve intermittent encounters, sparse vehicular networks can be considered examples of Delay/Disruption Tolerant Networks (DTNs). Closest in spirit to our work are two previous studies with an overlapping set of authors, who have examined the use of erasure codes in DTNs for reliably routing information between a particular source-destination pair [11], [12]. These studies provide a comparative analysis showing that the use of erasure coding can provide significant robustness to en-route path/node failures (the focus of [11]), as well as reduced latency (the focus of [12]), for push-based networks. In contrast, our emphasis in this work is on evaluating the latency and reliability of erasure coding for a pull-based network, specifically suitable for large files.

Also seemingly related to our work are papers that advocate the use of network coding for content dissemination or distribution. The use of network coding in the form of mixing of packets in intermediate nodes for content distribution was first proposed in the context of a content delivery system called Avalanche [23], [24]. Several researchers have extended this idea to adopt network coding to handle content distribution in vehicular networks, e.g., CodeTorrent [13], VANETCODE [25], CodeOn [26], and VCD [14]. Again, an essential distinction is that these works focus primarily on pushing files and messages to other nodes, whereas our focus is on pull-based retrieval for large files. The simpler pre-coded storage approach for file retrieval that we advocate is not network coding *per se* because it does not involve any in-network recombination of packets. Network coding, while intuitively appealing, is in fact not readily applicable to the setting we consider where individual nodes are seeking some particular content that is already stored in the network (not in the dynamic process of being disseminated). Dynamic network coding could still be (somewhat artificially, perhaps) introduced in our setting by forcing the continual transfer and re-coding of stored codewords whenever cars encounter each other; but this would give rise to a layer of additional complexity. This is because one would now have to decide the non-trivial question of which particular file's coded contents should be transferred and coded together when two cars encounter each other, and since storage is limited, one would also need to determine what other content should be evicted when new coded content is created. Network coding across the files might eliminate this issue, but will work only if the nodes were interested in all the files [27]. Our use of erasure coding avoids these problems entirely. For these reasons, we do not consider or evaluate dynamic network coding approaches in this work.

Finally, we note that our theoretical analysis relies on balls and bins processes (see e.g., [28]) and stochastic dominance arguments [29], [30] that are used to obtain bounds on the expected delay for coded storage.

3 Model and Problem Setup

In this section, we present a simplified model of a basic file sharing system and present a set of assumptions governing the model, making it amenable to analysis, but more importantly, giving us crucial insights into the system. Some of the simplifying assumptions (e.g., regarding mobility) will be relaxed later when we consider numerical simulations over realistic vehicular traces. We assume there are N identical participating seed vehicles (or nodes), each with a storage capacity of $\mathcal{C}$ bits allocated for the file sharing application. The total number of different files stored in the system is denoted by m; for simplicity, we assume that all the files have the same size of $\mathcal{M}$ bits (assume $\mathcal{C} \geq \mathcal{M}$) and are equally likely to be requested.

It is desired to distribute these m files to as many nodes as possible. It is assumed that the total available storage exceeds the total size of all files: i.e., $N\mathcal{C} \geq m\mathcal{M}$. Denote $\alpha = \frac{N\mathcal{C}}{m\mathcal{M}}$ and note that we can store each file $\alpha \geq 1$ times throughout the system and saturate the available capacity in the system. Typically, we will have $\alpha < N$, which means that each file will not be stored in all the nodes.

We refer to α as the **system redundancy**, since it is the number of times each bit is stored in the system. In this paper, we consider and analyze the expected delay in downloading files when uncoded replication scheme and the coded storage scheme are used. For the uncoded replication scheme (also called uncoded storage), we simply store each file α times in the nodes ensuring that a node doesn't store the same file multiple times (maximal spreading). On the other hand, for the coded storage scheme, an (n, k) MDS code is used and each file is split into k chunks and encoded into n chunks of the same size. We set $n/k = \alpha$, equal to the total system redundancy. This is because, as the effective size of each file after coding is $n\mathcal{M}/k$, in order to saturate the system capacity, we need $N\mathcal{C} = m(n\mathcal{M}/k)$, yielding $\alpha = n/k$.

We focus on the latency experienced by a given sink vehicle that is trying to download one of the m files. For the analysis, we assume an i.i.d. encounter model in which the sink is an external node that encounters any of the N nodes uniformly at random at each encounter. We impose a key communication constraint: whenever the sink meets any other vehicle, it can download at most d bits of data. We refer to d as the **bandwidth constraint**. Note that this parameter implicitly incorporates both the duration of the contact as well as the link rate. In our numerical simulations, we relax these simplifying assumptions, as we use encounters based on real vehicular traces and the download bandwidth is not a constant but rather a random variable that depends on the contact duration and the link quality model used.

Given all other parameters, we would like to determine the optimal values of n and k for coding. In order to do so, we note that each chunk has size $\mathcal{M}/k$ and so we want to choose k such that the chunk is downloadable within the bandwidth constraint (d). Thus we want $k \geq \mathcal{M}/d$, but since higher k equates to higher coding complexity, we use $k = \lceil \mathcal{M}/d \rceil$. The chunk size is therefore either $\mathcal{M}$ or d, whichever is lower (in practice we will have $d < \mathcal{M}$ for large files). Note that $k = 1$ in fact corresponds to not using any coding at all. Now, once k is fixed, choose $n = \alpha k$. Since there are N nodes, each node will contain $\beta = n/N$ chunks of a file. $\beta > 1$ implies there is at least one node which contains two different chunks for the same file.

We define the **delay** or **latency** D as the number of encounters needed before being able to fully reconstruct a file,

Variable	Brief Description
N	Number of nodes
m	Number of files
$\mathcal{C}$	Storage capacity of each node (bytes)
$\mathcal{M}$	File size (bytes)
d	Bandwidth limitation (bytes)
(n, k)	Coding parameters
α	Redundancy factor
β	Number of chunks from a file in the same node
D	Random variable denoting the delay in downloading a file

TABLE 1: List of common variables used.

and in the next section we quantify the expected latency $\mathbb{E}[D]$ for both the storage schemes. This can be multiplied by the expected inter-encounter time to give the latency in units of time.

We have listed the commonly used variables along with brief descriptions in Table 1.

4 Theoretical Analysis

Given the number of nodes N, the storage per node $\mathcal{C}$, the file size $\mathcal{M}$, and the number of files, we can compute the redundancy of each file $\alpha = \frac{N\mathcal{C}}{m\mathcal{M}}$. The files are stored according to coded or uncoded storage as described above, and the goal here is to analyze the expected delay in downloading a file from such a system. Since all files are equally popular, it is sufficient to consider any one file.

We make extensive use of balls and bins processes [28] in our analysis. The basic idea is to represent nodes as bins, and the throwing of a ball randomly into any of bins with equal probability models the sink meeting each node uniformly at random. The configuration of balls in bins that corresponds to a complete file download is defined differently in each case, as discussed below, but the common goal is to determine the expected time to reach this configuration, which corresponds to the expected delay. If instead of using uniform contact probabilities, even if we assume that the encounter probabilities of the sink with the nodes are non-uniform, the problem is equivalent to a balls and bins process with non-uniform bin selection probabilities and this problem is extremely hard with no known solutions.

4.1 Uncoded File Storage

We first analyze the latency of accessing a file in a vehicular network utilizing uncoded replication. Specifically, we show that the latency is inversely proportional to the redundancy in the system and the bandwidth constraint.

In this scheme, all the files are stored 'as such' in various nodes. We assume the system redundancy α to be an integer (recall that α is the number of times each file is stored in the system). Since the capacity $\mathcal{C} >$ file size $\mathcal{M}$, each file can be stored completely in a node. When the sink meets a node, it can download a maximum of d bits or $\mathcal{M}$ bits (entire file) whichever is lower. So depending on the values of d and $\mathcal{M}$, we can have two cases. If $d \geq \mathcal{M}$, then there is no bandwidth constraint at all. So, $\mathbb{P}[\text{a node is } \textit{good}] = \alpha/N$, where a good node is one which contains the required file. Thus the number of nodes to be seen before encountering a good node is a geometric random variable with mean $\mathbb{E}[D] = \frac{N}{\alpha}$. But if $d < \mathcal{M}$, only a fraction $d/\mathcal{M}$ of the file will be downloaded every time the sink meets a node. So $\mathbb{E}[D] = \frac{N}{\alpha}\left(\frac{\mathcal{M}}{d}\right)$.

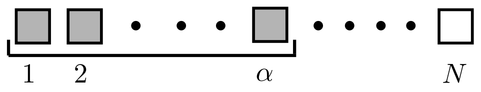


Fig. 1: A vehicular network with N nodes and redundancy α represented in the balls and bins framework. Each node is represented as a square, with the shaded squares containing copies of the file the sink is interested in.

Alternatively, in the balls and bins framework of Fig 1, this corresponds to throwing balls into N bins where each ball can land into any one bin with equal probability. If there is no bandwidth restriction, we are interested in counting the average number of balls to be thrown before a ball lands into one of the shaded bins (which is N/α as above). But when there is bandwidth restriction, we want to determine the number of balls in expectation that must be thrown until the first α bins contain $\mathcal{M}/d$ balls total. In this case, once a ball lands in any of the shaded bins, we repeat the experiment again. Note that a ball can fall into the same bin multiple times, which is equivalent to meeting the same node multiple times; but at each time the sink can download a different portion of the file. So $\mathbb{E}[D] = \frac{N}{\alpha}\left(\frac{\mathcal{M}}{d}\right)$ which is the same as that obtained above. Thus we have,

$$\mathbb{E}[D] = \begin{cases} N/\alpha & \text{if } d \geq \mathcal{M} \text{ or } \mathcal{M}/d \leq 1, \\ \frac{N}{\alpha}\left(\frac{\mathcal{M}}{d}\right) & \text{else if } d < \mathcal{M} \text{ or } \mathcal{M}/d > 1. \end{cases}$$

Combining,

$$\mathbb{E}[D] = \frac{N}{\alpha}\max(1, \mathcal{M}/d). \tag{1}$$

Hence, the expected delay is increased by a factor of $\mathcal{M}/d$ when there is a bandwidth constraint.

4.2 Coded File Storage

In this section, we analyze the expected delay in reconstructing a file under a coded storage scheme. As explained before, when using a (n, k) coding, each file is split into k chunks and then coded into n chunks and distributed to the nodes. In order to reconstruct the file, the sink has to download any k out of the n chunks. Whereas each file is stored α times in uncoded replication, it is expanded $\alpha = n/k$ times when using coding. Thus, analyzing the delays of both cases for the same α makes a fair comparison.

Balls and Bins model

Recall that β is the number of chunks per file that a node gets to store ($\beta = n/N$). As before, each node can be represented as a bin and thus we have N bins. Balls thrown into the bins are equivalent to the sink meeting a node at each time step. Whenever a sink meets a node, it can only download a single chunk since the chunk size is equal to the bandwidth constraint, and so the sink can meet the same node β times before running out of new data. Thus we can set the capacity

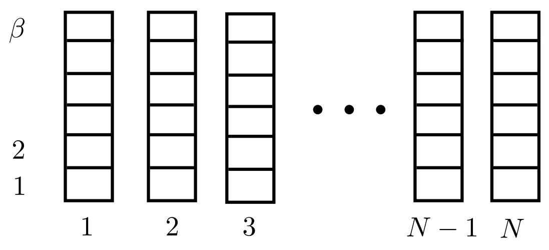


Fig. 2: The balls and bins model for the coded case for integer $\beta \geq 1$

of each bin to be β balls (assume β to be an integer). See Fig 2. In order to relax the condition on the integrality of β we note that we can set the capacity of a few bins to $\lceil\beta\rceil$ and the rest to $\lfloor\beta\rfloor$, making the bins non-identical and thus difficult to analyze. Also note that the final expression obtained does not require β to be an integer. We will note below what happens when $\beta < 1$.

Now, in order to get a file, we need to download any k different chunks out of the n chunks. *In the balls and bins process, we are interested in finding out the number of balls to be thrown in expectation, so that there are k total balls in all of the bins.* We make a note that since the bins have limited capacity, they could overflow and hence the required expectation is not always k. Let us analyze the delay by considering three different cases based on whether $\beta \leq 1$, $1 < \beta < k$ or $\beta \geq k$.

Case I: $\beta \leq 1$

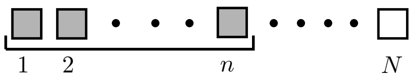


Fig. 3: The balls and bins set up when $\beta \leq 1$

In order to understand the capacity β being less than 1, we note that this in fact corresponds to $n \leq N$, since $\beta = n/N$. Thus only n out of the N nodes store the chunks. Without loss of generality, consider the first n nodes to contain the chunks. In the balls and bins process, we are interested in counting the number of balls in expectation to be thrown until there are k balls in any of the first n bins (see Fig 3). The expected number of balls to be thrown before the first ball lands into any of the n bins is N/n; the second ball takes $N/(n-1)$ in expectation and so on. Thus,

$$\begin{aligned}\mathbb{E}[D] &= \frac{N}{n} + \frac{N}{n-1} + \ldots + \frac{N}{n-k+1} \\ &= N(H_n - H_{n-k}) \approx N \log\left[n/(n-k)\right],\end{aligned}$$

since $H_n \approx \log n$. Using $\alpha = n/k$, we get,

$$\mathbb{E}[D] \approx N \log\left[\alpha/(\alpha-1)\right]. \tag{2}$$

Even though this equation does not depend on parameters like $\mathcal{M}$, $\mathcal{C}$ etc., there is an implicit dependence, since for example, α depends on N, $\mathcal{C}$, $\mathcal{M}$, m and we need to have the chunk size $\mathcal{M}/k$ to be equal to the bandwidth constraint d.

Case II: $1 < \beta < k$

To recall, β is the capacity per bin, or the number of times the same node can be seen before the sink runs out of useful chunks. Let us assume that β is an integer. We are interested in finding out the expected number of throws to get k balls into the system. Deriving an exact expression for $\mathbb{E}[D]$ seems hard and so we upper bound the expected delay. Also note that $\mathbb{E}[D] \geq k$.

Let us define the state of the system $\mathcal{S}$ at any time as the arrangement of the balls in the bins and $|\mathcal{S}|$ to be the number of balls in the system. For example when $|\mathcal{S}| = 2$, valid states include $\mathcal{S} = \{2, 0, 0, \ldots, 0\}, \{1, 1, 0, \ldots, 0\}$ etc., where the j^{th} element in the set corresponds to the number of balls in the j^{th} bin. For a general i, there are an exponential number of states $\mathcal{S}$ such that $|\mathcal{S}| = i$.

Let $T_{i\to i+1}$ be the number of balls required to add one more ball to the system, given that there are already i balls in the system. The expected delay is then

$$\mathbb{E}[D] = \sum_{i=0}^{k-1} \mathbb{E}[T_{i\to i+1}]. \tag{3}$$

We first note that the distribution of $T_{i\to i+1}$ can be determined if the current state is given, otherwise it is extremely difficult. For example, given that $\mathcal{S} = \{0, 0, \ldots, 0\}$ (i.e. $i = 0$ and there are no balls in the system), $T_{0\to 1}$ is 1 with probability 1 (or geometric with failure probability 0); and given that the state is $\mathcal{S} = \{\beta, 0, 0, \ldots, 0\}$ (i.e. the first bin is full with β balls), then $T_{\beta\to\beta+1}$ is geometric with failure probability $1/N$. Thus once we know the state, we can determine the distribution of $T_{i\to i+1}$. But what can we say about $T_{i\to i+1}$ without conditioning on the state? As an example, suppose there are $i = \beta$ balls in the system, then there is a finite probability q with which one of the bins may be full, in which case, the distribution is geometric with failure probability $1/N$ and with the remaining probability $(1-q)$, the distribution is geometric with failure probability 0. Thus we can express $\mathbb{P}[T_{\beta\to(\beta+1)} = z] = q\mathbb{P}[\text{Geom}(1/N) = z] + (1-q)\mathbb{P}[\text{Geom}(0) = z]$, where $\text{Geom}(x)$ is a geometric random variable with failure probability x. We make a note that $T_{\beta\to(\beta+1)}$ is a probabilistic mixture of two geometric random variables with mixing probabilities q and $1-q$. For a general i, $T_{i\to i+1}$ is a probabilistic mixture of at most N geometric random variables. Even though it is difficult to determine the mixing probabilities for every i, we can effectively eliminate them, for which we use the concept of Stochastic dominance (see [29], [30] for more details):

Defn: Stochastic Dominance Consider two random variables X and Y, possibly defined on different probability spaces. When X is stochastically smaller than Y, then for every $z \in \mathbb{R}$, the probability inequality $\mathbb{P}(X \leq z) \geq \mathbb{P}(Y \leq z)$ must hold or in terms of the cumulative distribution function, $F_X(z) \geq F_Y(z)$. This is denoted as $X \preceq Y$, i.e. X is stochastically dominated by Y.

Another concept we need below is that of Coupling.

Defn: Coupling For a given set of random variables $X_1, X_2, \ldots, X_n$, a coupling is defined as a new set of random variables $(\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n)$ over the same probability space such that the marginal distribution of $\hat{X}_i$ is same as that of X_i for $i = 1, 2, \ldots, n$. Thus for all measurable subsets E of $\mathbb{R}$, $\mathbb{P}(\hat{X} \in E) = \mathbb{P}(X \in E)$.

Remark If $X \preceq Y$, then $\mathbb{E}[X] \leq \mathbb{E}[Y]$. This is noted by seeing that $\mathbb{E}[X] = \sum_z (1 - F_X(z)) \leq \sum_z (1 - F_Y(z)) = \mathbb{E}[Y]$.

We list three useful lemmas below.

Lemma 4.1: A random variable X is stochastically dominated by another random variable Y if and only if there exists a coupling $(\hat{X}, \hat{Y})$ of X and Y such that $\mathbb{P}(\hat{X} \leq \hat{Y}) = 1$. Refer to Lemma 2.11 in [30] for proof.

Lemma 4.2: Let $X \sim \text{Geom}(p)$ and $Y \sim \text{Geom}(q)$, where p and q are the failure probabilities. If $p \leq q$, then $X \preceq Y$.

Proof: At each step (starting from 0), a real number is selected at random from $[0,1]$ and $\hat{X}$ and $\hat{Y}$ are defined to denote the step at which the number chosen belongs outside $[0,p]$ or $[0,q]$ respectively. We note that if $\hat{X}$ succeeds at step x, so that the number chosen at that step is for the first time higher than p, then $\hat{Y}$ could not have succeeded before or at x i.e. $\mathbb{P}(\hat{Y} \geq x \mid \hat{X} = x) = 1$. $\mathbb{P}(\hat{X} \leq \hat{Y}) = \sum_0^\infty \mathbb{P}(\hat{Y} \geq x \mid \hat{X} = x)\mathbb{P}(\hat{X} = x) = \sum_0^\infty (1)\mathbb{P}(\hat{X} = x) = 1$. Thus $\mathbb{P}(\hat{X} \leq \hat{Y}) = 1$ giving $X \preceq Y$. Hence a geometric random variable is always dominated by another geometric random variable with higher failure probability. □

Lemma 4.3: Let us have l random variables $X_1, X_2, \ldots, X_l$, with $X_j \preceq X_1$ for all $j = 2, 3, \ldots, l$. If X is a probability mixture of $X_1, X_2, \ldots, X_l$, such that $p_X(z) = \sum_{j=1}^l \alpha_j p_{X_j}(z)$ with constants $\alpha_j \geq 0$ $(j = 1, 2, \ldots, l)$ and $\sum_{j=1}^l \alpha_j = 1$, then $X \preceq X_1$.

Proof of Lemma 4.3: $F_X(z) = \sum_{j=1}^l \alpha_j F_{X_j}(z) \geq \sum_{j=1}^l \alpha_j F_{X_1}(z) = F_{X_1}(z)$. Thus we have $F_X(z) \geq F_{X_1}(z)$ for all z, which implies $X \preceq X_1$. □

In other words, this lemma states that a probabilistic mixture of geometric random variables is stochastically dominated by the constituent geometric random variable with the biggest failure probability.

Back to the case when $i = \beta$, since the biggest failure probability is $1/N$, the corresponding geometric random variable stochastically dominates other geometric random variables (Lemma 4.2), and so from Lemma 4.3, we can see that $T_{\beta \to \beta+1} \preceq \text{Geom}(1/N)$, conveniently removing the dependence on q. Thus we note that when $T_{i \to i+1}$ is a probabilistic mixture of geometric random variables with biggest failure probability p, then $T_{i \to i+1} \preceq \text{Geom}(p)$. We are now all set to get the upper bound on the latency.

Theorem 4.4: The expected delay due to coded storage in the case when $1 < \beta < k$ is upper bounded by $n(H_N - H_{\lceil N(1-1/\alpha) \rceil})$, where H_N is the N^{th} harmonic number.

Proof: Consider a random variable $D'_i = \sum_{i=0}^{k-1} T'_{i \to i+1}$ where $T'_{i \to i+1}$ is a geometric random variable with failure probability p'_i. By suitably choosing p'_i, we will first prove that $T_{i \to i+1}$ is stochastically dominated by $T'_{i \to i+1}$ for each i.

When the first β balls are thrown, none of the bins could have overflowed and so $T_{i \to i+1}$ is geometric with failure probability 0 for $i = 0, 1, \ldots, \beta - 2, \beta - 1$. We choose $p'_i = 0$ in these cases. Once there are β balls in the system, as explained above, $T_{i \to i+1}$ is a probabilistic mixture of geometric random variables with the biggest failure probability $1/N$. From the insight above, we set $p'_\beta = 1/N$, so that $T_{i \to i+1} \preceq \text{Geom}(1/N)$. Also, when $\beta \leq i \leq 2\beta - 1$, its not possible to have two or more bins full, and so in all these cases, the biggest failure probability is $1/N$; thus we set $p'_i = 1/N$ for $\beta \leq i \leq 2\beta - 1$. Further, its not difficult to see that we should set $p'_{2\beta} = 2/N$.

Using similar arguments, we set $p'_i = j/N$ for $j\beta \leq i < (j+1)\beta$, for $j = 0, 1, \ldots, (k/\beta - 1)$ (assuming $k/\beta = x$ to be an integer, otherwise use $x = \lfloor k/\beta \rfloor$ above). For the last case when $i = k - 1$, since x or more bins cannot be full, set $p'_{k-1} = (x-1)/N$. We note that $x = \frac{k}{\beta} = \frac{k}{n}N = \frac{N}{\alpha}$.

For each i, since we have chosen p'_i to be at least as big as the biggest failure probability in the probabilistic mixture of geometric random variables that constitute $T_{i \to i+1}$, we can note that $T_{i \to i+1} \preceq \text{Geom}(p'_i)$ using Lemmas 4.2 and 4.3. This implies $\mathbb{E}[T_{i \to i+1}] \leq \mathbb{E}[\text{Geom}(p'_i)]$ and thus $\mathbb{E}[D] \leq \mathbb{E}[D']$ by summing over all i. Noting that $\mathbb{E}[\text{Geom}(p)] = \frac{1}{1-p}$,

$$\begin{aligned}
\mathbb{E}[D'] = \sum_{i=0}^{k-1} \mathbb{E}[T'_{i \to i+1}] &= \sum_{j=0}^{x-1} \sum_{i=j\beta}^{(j+1)\beta - 1} \mathbb{E}[T'_{i \to i+1}] \\
&= \sum_{j=0}^{x-1} \sum_{i=j\beta}^{(j+1)\beta - 1} \mathbb{E}[\text{Geom}(j/N)] = \sum_{j=0}^{x-1} \frac{\beta}{1 - j/N} \\
&= N\beta(H_N - H_{N-x}) = n(H_N - H_{\lceil N(1-1/\alpha) \rceil}).
\end{aligned}$$

Since $k \leq \mathbb{E}[D] \leq \mathbb{E}[D']$, we get

$$k \leq \mathbb{E}[D] \leq n(H_N - H_{\lceil N(1-1/\alpha) \rceil}).$$

□

Note that when using this expression, it does not matter whether x or β is an integer or not. Also, for high values of α, $n(H_N - H_{\lceil N(1-1/\alpha) \rceil}) \approx \beta N \log[\alpha/(\alpha - 1)]$ since $\beta = nN$.

The above analysis is not restricted to keeping $\beta < k$. We can note that when $\beta \geq k$, $\alpha = n/k \geq N$ and thus for large values of α, the average delay expression can be approximated as $k \leq \mathbb{E}[D] \leq k[1 + 1/2\alpha + o(1/\alpha^2)]$, thus confirming with the result below.

Case III: $\beta \geq k$

Since the capacity is sufficiently large, no bin can get full in k throws and so $\mathbb{E}[D] = k$.

To summarize,

$$\mathbb{E}[D] \begin{cases} \approx N \log\left(\frac{\alpha}{\alpha-1}\right) & \text{if } \beta \leq 1 \\ \leq \beta N \log\left(\frac{\alpha}{\alpha-1}\right) & \text{else if } \beta > 1 \end{cases}$$

Combining, we obtain our final bound,

$$\mathbb{E}[D] \leq \max(1, \beta) N \log\left(\frac{\alpha}{\alpha - 1}\right) \approx \frac{N}{\alpha} \max\left(1, \frac{\mathcal{M}}{d}\frac{\alpha}{N}\right), \tag{4}$$

where the approximation holds for high values of α.

A note on the choice of k: in order to derive all the above expressions, we have chosen $k = \mathcal{M}/d$. But what would happen if the k is chosen any higher? First, consider the case when n is a multiple of N. By choosing k twice its actual value, for instance, each node will have only half the chunk size as before, but twice the number of chunks, so the amount of data per node is the same, thus the average delay will be the same. Now if $n < N$, by increasing k, we also increase n, i.e. more nodes contain desired data but less of it. Overall, we did not observe any improvement in the average delay by increasing k.

4.3 The Benefits of Coding: Summary

By comparing eqn (1) and eqn (4), it is clear that the expected delay with coding is at least as good or better than uncoded replication. The interesting cases of d are when $\mathcal{M}/d = 1$ and $\frac{\mathcal{M}}{d}\frac{\alpha}{N} = 1$; the former giving $d = \mathcal{M}$ and the latter giving $d = \frac{\mathcal{M}\alpha}{N} = \mathcal{C}/m$. Thus we have the following three regimes:

- *(High Bandwidth regime)* $d \geq \mathcal{M}$: The expressions for the latencies in both the coded and uncoded storage schemes become almost equal with $\mathbb{E}[D_{\text{uncoded}}] = N/\alpha \approx \mathbb{E}[D_{\text{coded}}]$, and so coding performs same as uncoded replication (note that since $k = 1$, coding is equivalent to uncoded file replication).
- *(Intermediate Bandwidth regime)* $\mathcal{C}/m \leq d \leq \mathcal{M}$: From the expressions, we have $\mathbb{E}[D_{\text{uncoded}}] = \frac{N}{\alpha}\frac{\mathcal{M}}{d}$ and $\mathbb{E}[D_{\text{coded}}] \leq \frac{N}{\alpha}$. Thus the improvement of using coding is $\mathcal{M}/d$. Each node cannot store chunks from all the files due to $\mathcal{C}/m \leq d$ and so the sink has to wait to meet good nodes, which is the only factor contributing to the delay.
- *(Bandwidth limited regime)* When $d \leq \mathcal{C}/m$, we obtain $\mathbb{E}[D_{\text{coded}}] \leq \frac{\mathcal{M}}{d}$ (and since $\mathbb{E}[D_{\text{coded}}] \geq k = \frac{\mathcal{M}}{d}$, we have that $\mathbb{E}[D_{\text{coded}}] = \frac{\mathcal{M}}{d}$). Because, $\mathbb{E}[D_{\text{uncoded}}] = \frac{N}{\alpha}\frac{\mathcal{M}}{d}$, the improvement here is N/α. Thus under such a severe bandwidth constraint, coding performs as if complete files were available in all the nodes, only to be limited by the bandwidth.

5 Trace Based Experiments

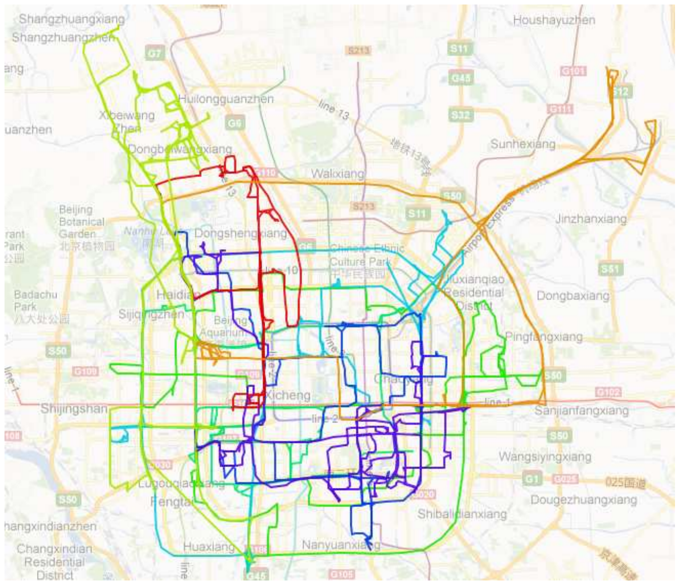
We now turn to an empirical evaluation of the benefits of coded storage, using real vehicular traces. We use GPS traces of 1,000 taxis in Beijing and 1,608 buses in Chicago.

We assume that the nodes continue to run their application throughout the day. Note that we do not assume the nodes to be moving throughout the day, only that on-board radio and the computer may continue to work even if the node is stopped.

For inter-vehicular communication, we used a realistic model of IEEE 802.11p from [31], the details of which are given in section 5.4 below. We next present the description of the datasets used.

5.1 Dataset Description

The Beijing dataset consists of GPS traces collected from 00:00hrs to 23:59hrs on Jan 5, 2009 local time, recorded every minute for a total of 2,927 taxis. The GPS co-ordinates span 32.1223 to 42.7413 in latitude and 111.6586 to 126.1551 in longitude. Of these 2,927 taxis, we chose a thousand randomly


(a) Beijing dataset

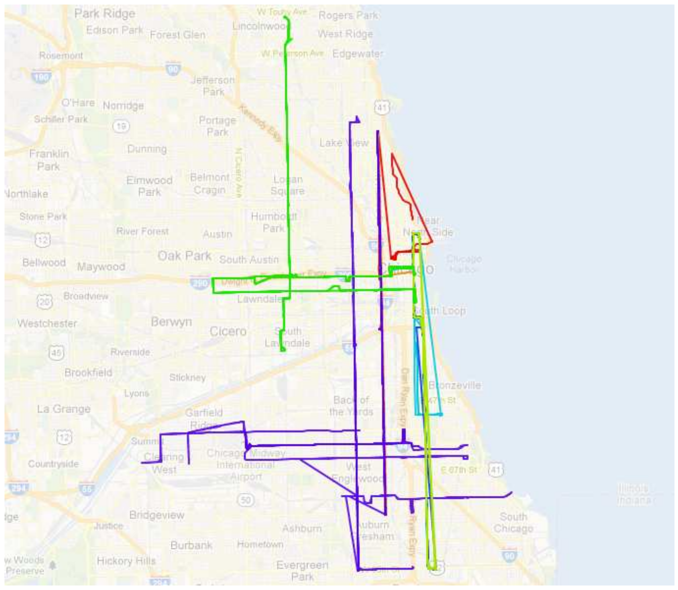
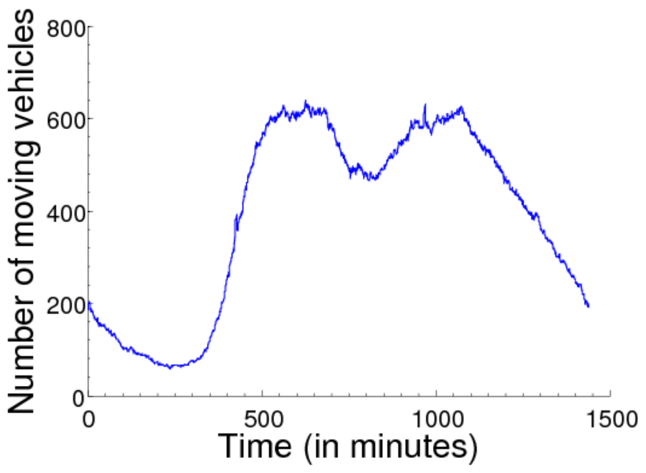

(b) Chicago dataset

Fig. 4: Maps of the routes traced by a few randomly selected nodes in the Beijing and the Chicago datasets. We limited the number of nodes so as to not clutter the image. Colors are chosen randomly for each node by the tool we used to plot the routes.

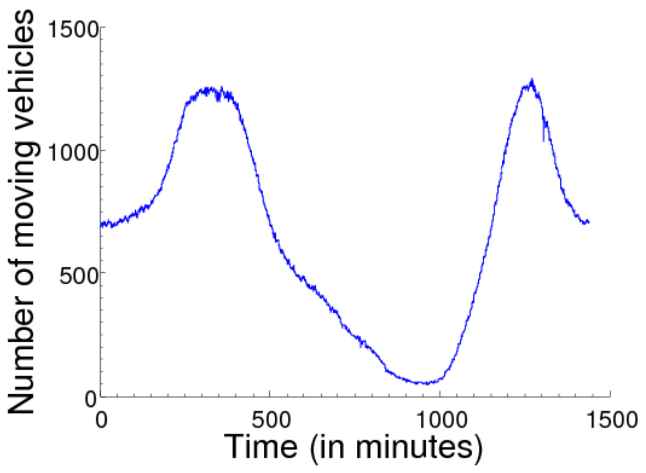
for our simulations. In Fig 4a, we show[1] the routes taken by a randomly chosen subset of these thousand taxis. Note that we used about 8 taxis to display to avoid clutter.

For the Chicago dataset, we collected data starting from Nov 1, 2010 at 11:06hrs (Chicago local time) for every 30 seconds and used data worth the first 24hrs. The latitudes and longitudes of this dataset range from 41.6440 to 42.0651 and -87.8866 to -87.5256 respectively. The routes taken by a random subset of these nodes (7 buses) are shown in Fig 4b. Since the routes are carefully planned ahead, one can see the

1. We used gpsvisualizer.com and google.com to obtain these plots.



(a) Beijing dataset



(b) Chicago dataset

Fig. 5: Density of moving taxis vs time

difference between the routes in Fig 4a and Fig 4b. We also note that many routes overlap with each other (spatially if not temporally) and so not all can be seen clearly from the map.

In Fig 5, the density of the nodes is shown for each dataset, for which we plot the number of moving nodes versus time. In Fig 5a and Fig 5b, the 0 minute corresponds to the time the dataset begins, and hence for the Beijing dataset it is 00:00 hrs, whereas for the Chicago dataset it is 11:06 hrs (both local time). We determine whether a node is not moving if its coordinates do not change for a continued duration (about two minutes). We note that the average duration that a taxi is moving is 9.4 hours in the Beijing dataset and quite remarkably, this value is 9.6 hours for the Chicago dataset.

It can be seen in Fig 5a that as the data set starts at 12am, the density drops to the lowest at around 4am and starts to pick up and reaches a peak between 8am and 10am. It drops after that but again reaches a peak between 4pm and 6pm, after which it starts decreasing rapidly. Correspondingly in Fig 5b, which starts at about 11am, the density peaks between 4pm and 6pm, then drops to very low at around 4am and then picks up again to reach a peak at 8am.

5.2 Performance Metrics

In order to characterize the performance of the system, we cannot simply use the average delay in downloading a file as a figure of merit. This is because, since the traces are time limited, there could be files that may not get fully reconstructed by the end of the duration of the trace, and so it is hard to quantify the delay of such incompletely downloaded files. Thus, we rely primarily on two metrics: one is the *full-recovery probability*, which measures the probability that a file can be fully recovered by a sink by a given time and the other is the *average file download percentage*, which measures, on average, how much of a file is downloaded by a given time. Thus, for example, a file-recovery probability of 0.9 means that the nodes were able to successfully download full files 90% of the time and an average file download percentage of, say, 95 means that the nodes were able to download 95% of the file on the average.

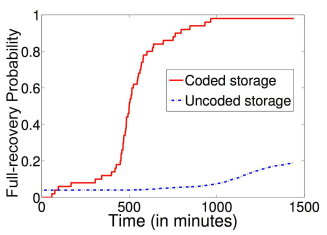
5.3 Experiment Methodology

The nodes are indexed from 1 to N, where N is the number of nodes ($N = 1000$ for the Beijing dataset, and $N = 1608$ for the Chicago dataset). The files are indexed 1 through m. Since the end goal is to deploy a file sharing system in a vehicular network, we try to make reasonable choices of various parameters involved. A capacity of 100GB per node is assumed as a default, unless specified otherwise. Similarly, by default, files are assumed to be of size 1GB, typical of movie clips and we consider a default of 2,500 files in the system.

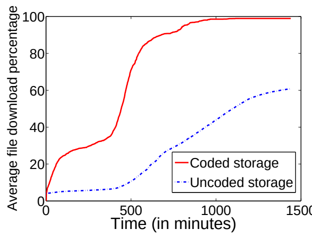
As explained before, the two primary metrics of performance are the full-recovery probability and the average file download percentage, both characterized as functions of time. Therefore, our experimental methodology is to carry out a number of experiments, and in each there is a sink trying to download a file. We record these metrics of interest along time and average across experiments.

Each experiment consists of the following three steps: first, the files are allocated to the nodes; then, a sink-file pair is determined; and this is followed by a simulation of the encounter between the sink and the rest of the nodes using the trace. We will describe in detail these aspects next.

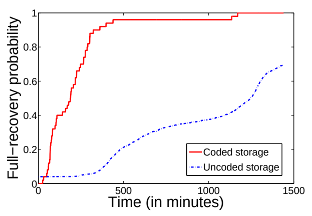
The first step is that of storing the files onto the nodes. If coding is not used, files are not transformed; but if coding is used, files are encoded to get chunks. Both in the uncoded and the coded storage schemes, the files and the chunks are stored by ensuring maximal spreading. That is, in the case of uncoded storage, we make sure to not store the same file twice or more in the same node; and in the case of coded storage, multiple chunks of the same file are not stored in the same node, unless all other nodes have been used. In fact, we found that by randomly storing files/chunks, coding still performed virtually the same whereas the performance of uncoded storage scheme decreased slightly. Thus, we decided to use maximal spreading so as not to worry about the performance degradation introduced by randomization, even though random storage may be more realistic.



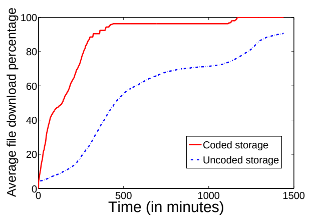
(a) Full-recovery probability (Beijing)



(b) Average file download percentage (Beijing)



(c) Full-recovery probability (Chicago)



(d) Average file download percentage (Chicago)

Fig. 6: Evaluating the performance of distributed storage codes in the default setting consisting of 2,500 files each of size 1GB stored in nodes each having 100GB storage for both the Beijing and Chicago datasets. There are 1,000 nodes in total in the Beijing dataset and 1,608 nodes in the Chicago dataset.

Next, a random node is selected to be a sink by selecting a random index from $\{1, 2, \ldots, N\}$, and it tries to download a random file (by choosing a random index from $\{1, 2, \ldots, m\}$).

The third step involves the simulation of the contacts between the sink and the other nodes, so that the sink can download the chosen file opportunistically. Note that the day-long trace is divided into intervals of length one minute each for the Beijing dataset and 30 seconds each for the Chicago dataset, resulting in a total of 1440 and 2880 slots for the Beijing and the Chicago dataset respectively. The choice of the granularity is dictated by the dataset. At each slot, we determine the distance between the given sink and every other node, and apply the radio model (described below) to find out the number of packets transferred, if any.

For each experiment, we keep track of the percentage of the file downloaded and whether the file download is complete or not at each time step. When presenting the results, we average over 50 random sinks, and for each sink, we run the entire simulation 100 times choosing a different file each time.

5.4 Realistic Radio Link Model

The IEEE 802.11p standard specifies the data rate to range from 1.5Mbps to 27Mbps with the default being 3Mbps, which we use in our simulations. For inter-vehicular communication, we use an empirical model of packet delivery characteristics obtained from [31]. The authors characterize the packet delivery ratio (PDR) against various parameters such as the separation between two nodes, their relative velocity etc., in a number of different environments and the overall experiments lasted for about 30 hours. Of the various environments in which their experiments were conducted, the closest match to our dataset is the Suburban Road (SR) environment. Thus we use their PDR vs separation distance data (Fig 3(a) in [31]) to carry out our simulations. It may also be emphasized that the authors found that the relative velocity between two nodes does not significantly affect the PDR, the way inter-vehicular distance does. We choose packet sizes of 380 bytes with payload 300 bytes. Additionally a protocol set up time of about 1ms is considered (based on [31]).

5.5 Choice of the coding parameter k

From the Beijing dataset, we observed an average contact duration of 55.6s (assuming a radio range of 500m) leading to an average data transfer of 21MB (at 3Mbps under ideal conditions). Since it is desirable to be able to transfer multiple chunks per encounter, we choose a safe chunk size of 1MB. We use this same chunk size for the Chicago dataset too.

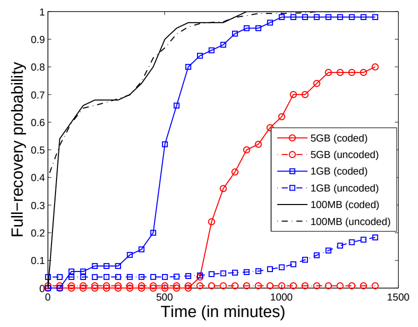
5.6 Discussion of the Results

Our most important results are shown in Fig 6, in which we consider a typical file sharing scenario with 2,500 files each of size 1GB; and each node having about 100GB storage. Such a system is implemented atop both the datasets, and both the full-recovery probability and the average file download percentage are measured for each time step. While we primarily discuss the results with respect to the Beijing dataset, similar discussions follow for the Chicago dataset. We note that coding offers significant benefits compared to uncoded replication. For example, at the end of 24 hours, files are reconstructed fully 98% of the time by using coding, whereas without coding, only 19% of the files are reconstructed fully (see Fig 6a). The corresponding values for average file download percentage are 99% and 61% respectively. If we were to consider the instant when 80% of nodes are able to complete their downloads, this corresponds to about 600 minutes in the trace when coding is used, but only 4.4% of nodes are successful in full downloads by 600 minutes if coding is not used. An interesting observation to make is that since the Beijing trace begins in the middle of the night with relatively little traffic, one can see from Fig 6 that the rate at which files are completed starts to slow down around 60 minutes (1 a.m.) and then picks up again at 400 minute (7 a.m.). No such trend can be seen in the Chicago dataset because the dataset starts at around 11am Chicago local time. Another factor affecting the rate towards the end is the scarcity of new chunks (similar to the coupon collector problem).

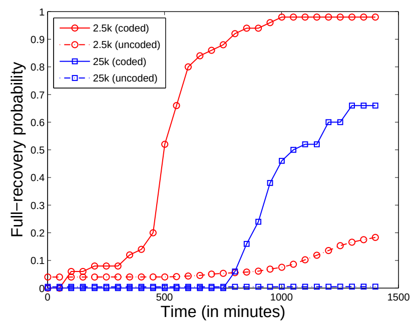
Further, we performed a number of experiments to thoroughly understand the effect of various parameters on the performance of the system, by systematically varying the parameters $\mathcal{M}$, $\mathcal{C}$ and m. In our evaluations, we keep two parameters constant and vary the third. The results are shown for the Beijing dataset and those of the Chicago dataset are omitted for brevity since they display similar trends.

5.6.1 *Effect of file size*

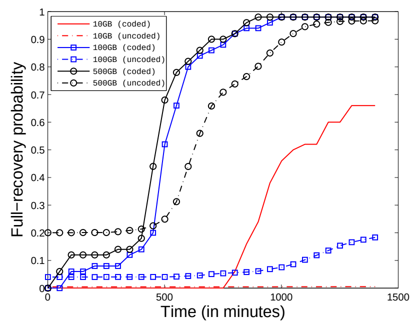
As file size increases, since system storage remains constant, we are effectively decreasing the system redundancy, which



(a) File size varied

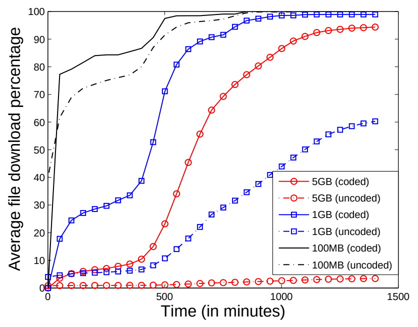


(b) Number of files varied

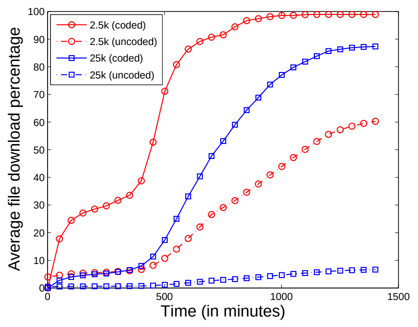


(c) Capacity per node varied

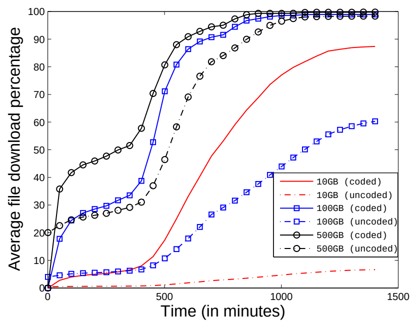
Fig. 7: Plots showing how various parameters affect the full-recovery probability. In each of the cases, one parameter is varied while keeping the others constant. Typical values used are a storage capacity of 100GB, 2,500 files and file size 1GB.



(a) File size varied



(b) Number of files varied



(c) Capacity per node varied

Fig. 8: Plots showing the impact of different parameters on the average file download percentage. The parameters are same as in Fig 7.

should adversely impact latency. This is observed for both coded and uncoded storage, but there are clear differences in relative performance. We notice from Fig 7a and Fig 8a that when the file size is very small (100MB in the figures), coding offers no benefit at all. But as the file size is increased to 1GB, coding offers tremendous improvements by being able to fully download full files most of the time (98% of the time in Fig 7a), whereas only about a fifth of the time (Fig 7a) without coding. When the file size is increased further to 5GB, the performance of coding suffers, but not drastically, whereas in the absence of coding, the probability of full recovery drops almost to zero (from Fig 8a, and we note that many sinks have been able to download about a tenth of the file on the average, but not a complete file).

5.6.2 *Effect of the number of files and the capacity*

Figs 7b and 8b show the impact of the number of files on the system performance. As the number of files increases, the system redundancy decreases and hence the full-recovery probabilities and the file download percentages both start to decrease. And, as the capacity increases from 10GB to 100GB to 500GB, files can be replicated many more times and hence the full-recovery probabilities and the file download percentages both start to get better (Fig 7c and Fig 8c). An interesting observation to make is that the curve corresponding to the case when there are 25,000 files with 100GB storage per car in Fig 7b and the curve corresponding to 2,500 files with 10GB storage per car in Fig 7c (or Fig 8b and Fig 8c) are both identical (if we choose the same set of sink file pairs). This is because having 25,000 files on nodes with 100GB has the same system redundancy as having 2,500 in 10GB nodes. Also note that some of the probabilities or percentages for the uncoded replication start non-zero, since some of the sinks already contain the files they are interested in, whereas when coding is used, no node can contain a full file by itself and so all the probabilities and percentages are 0 to begin with.

5.7 Absolute File Download Latency

A cautionary note is in order in interpreting our results in this section in terms of the absolute numbers, which suggest that downloading a large 1GB-sized file in a vehicular network is likely to take six to ten hours even with coding. We note that our traces, though they involve in the order of 1,000 nodes, are still relatively quite sparse in terms of encounters as they involve large areas in Beijing and Chicago. Thus the latency values presented in our study in terms of absolute numbers may not be representative of what might be possible with much denser vehicular network deployments (say 100,000+ vehicles in a large city) during high-traffic hours. But the dramatic gaps observed between the performance of coded and uncoded storage in these simulations indicate strongly that the use of coding is essential for speeding up large file downloads in encounter-based vehicular networks, regardless of vehicular density.

6 Conclusion

We have studied the effect of coded storage on the latency of on-demand, pull-based content access in an intermittently connected vehicular network. We developed a mathematical model to study the relative benefits, and proved that optimized coded storage is never worse than uncoded storage, and can significantly improve the latency performance in the case of large files and bandwidth limitations. We have further validated our findings using realistic simulations based on large-scale vehicular trace involving taxis in Beijing and buses in Chicago. Our numerical results confirm that file download latency (particularly for large files) is improved dramatically when the content is stored using erasure codes.

There are still many unanswered questions. Open questions include how to re-distribute content when there is node churn, and the possibility of learning patterns in vehicular encounters to further optimize the content storage. Another interesting problem is to determine the optimal storage strategy if the popularities of various files are known beforehand. Some of the preliminary results we have on unequal file popularities are available in [32]. We also note that our traces, albeit involving 1,000 cars, are still relatively sparse given that they involve city-scale mobility. It is important to investigate content access latency in denser deployments, to understand the performance of file sharing in large-scale vehicular networks.

References

[1] D. Jiang and L. Delgrossi, “IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments,” in *IEEE VTC Spring 2008*.

[2] L. Armstrong, “Classes of Applications,” *Presentation, http://tinyurl.com/vanetapps*.

[3] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, “The pothole patrol: Using a mobile sensor network for road surface monitoring,” in *Proc. of ACM MobiSys*, 2008.

[4] A. Nandan, S. Tewari, S. Das, M. Gerla, and L. Kleinrock, “AdTorrent: Delivering location cognizant advertisements to car networks,” in *Proc. IEEE/IFIP WONS*, 2006.

[5] K. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla, “First experience with cartorrent in a real vehicular ad hoc network testbed,” in *IEEE MOVE*, May 2007.

[6] U. Lee, J. Lee, J. Park, and M. Gerla, “Fleanet: A virtual market place on vehicular networks,” *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 1, pp. 344–355, 2010.

[7] S. Kapadia, B. Krishnamachari, and S. Ghandeharizadeh, “Static Replication Strategies for Content Availability in Vehicular Ad-hoc Networks,” *Mobile Networks and Applications*, vol. 14, no. 5, pp. 590–610, 2009.

[8] “Excited About the Cloud? Get Ready for Capped Data Plans,” *http://pogue.blogs.nytimes.com/2011/06/16/excited-about-the-cloud-get-ready-for-capped-data-plans/*.

[9] A. Vahdat and D. Becker, “Epidemic routing for partially connected ad hoc networks,” CS-2000-06, Duke University, Tech. Rep., 2000.

[10] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, “Spray and wait: an efficient routing scheme for intermittently connected mobile networks,” in *WDTN*, 2005.

[11] S. Jain, M. Demmer, R. Patra, and K. Fall, “Using redundancy to cope with failures in a delay tolerant network,” *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 109–120, 2005.

[12] Y. Wang, S. Jain, M. Martonosi, and K. Fall, “Erasure-coding based routing for opportunistic networks,” in *Proc. of ACM WDTN*, 2005.

[13] U. Lee, J. Park, J. Yeh, G. Pau, and M. Gerla, “CodeTorrent: Content Distribution using Network Coding in VANET,” in *Proc. of ACM MobiShare*, 2006.

[14] U. Shevade, Y.-C. Chen, L. Qiu, Y. Zhang, V. Chandar, M. K. Han, H. H. Song, and Y. Seung, “Enabling high-bandwidth vehicular content distribution,” in *Proc. of ACM CoNEXT*, 2010.

[15] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[16] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” in *Proc. of ACM SIGCOMM*, 1998.

[17] M. Luby, “LT Codes,” *Proc. of IEEE Foundations of Computer Science*, 2002.

[18] A. Shokrollahi, “Raptor codes,” *IEEE Trans. on Information Theory*, June 2006.

[19] A. Dimakis, V. Prabhakaran, and K. Ramchandran, “Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes,” in *IPSN*, 2005.

[20] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein, “Growth codes: Maximizing sensor network data persistence,” *Proc. of ACM SIGCOMM*, 2006.

[21] H. Weatherspoon and J. D. Kubiatowicz, “Erasure coding vs. replication: a quantitiative comparison,” in *Proc. of IPTPS*, 2002.

[22] R. Rodrigues and B. Liskov, “High availability in DHTs: Erasure coding vs. replication,” in *Proc. of IPTPS*, 2005.

[23] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *Proc. of IEEE Infocom*, 2005.

[24] C. Gkantsidis, J. Miller, and P. Rodriguez, “Comprehensive view of a live network coding P2P system,” in *Proc. of ACM IMC*, 2006.

[25] S. Ahmed and S. Kanhere, “VANETCODE: Network Coding to enhance cooperative downloading in Vehicular Ad-hoc Networks,” in *Proc. of ACM ICWCMC*, 2006.

[26] M. Li, Z. Yang, and W. Lou, “CodeOn: Cooperative popular content distribution for vehicular networks using symbol level network coding,” *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 1, pp. 223–235, 2011.

[27] B. Haeupler, “Analyzing network coding gossip made easy,” *Arxiv preprint arXiv:1010.0558*, 2010.

[28] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge Univ Press, 2005.

[29] G. Grimmett and D. Stirzaker, *Probability and random processes*. Oxford University Press, USA, 2001.

[30] R. Van Der Hofstad, “Random graphs and complex networks,” *Available on http://www.win.tue.nl/˜rhofstad/NotesRGCN.pdf*, 2009.

[31] F. Bai, D. Stancil, and H. Krishnan, “Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers,” in *Proc. of ACM MobiCom*, 2010.

[32] M. Sathiamoorthy, A. G. Dimakis, B. Krishnamachari, and F. Bai, “Distributed Storage Codes Reduce Latency in Vehicular Networks,” CENG Tech Report, CENG-2011-3, available at http://ceng.usc.edu/assets/003/75382.pdf, Tech. Rep., 2011.

Maheswaran Sathiamoorthy is a PhD Candidate at the University of Southern California. He received his bachelors from the Indian Institute of Technology Kharagpur, India in 2008. He was an Annenberg Fellow from 2008-2012. His research interests are in distributed storage for vehicular network clouds and data center clouds.

Alexandros G. Dimakis is an Assistant Professor at the Viterbi School of Engineering, University of Southern California since 2009. He currently holds the Colleen and Roberto Padovani Early Career Chair in Electrical Engineering. He received his Ph.D. in 2008 and M.S. degree in 2005 in electrical engineering and computer sciences from UC Berkeley and the Diploma degree in Electrical and Computer Engineering from the National Technical University of Athens in 2003. During 2009 he was a postdoctoral scholar at the CMI center at Caltech.

He received the NSF Career award in 2011, the Eli Jury dissertation award in 2008. He is the co-recipient of several best paper awards including the the Joint Information Theory and Communications Society Best Paper Award in 2012.

He is currently serving as an associate editor for IEEE Signal processing letters and was the chair of the Data Storage track at Globecom and a keynote speaker at the Int. Symposium on Network Coding (NetCod). His research interests include communications, coding theory, signal processing, and networking, with a current focus on distributed storage, network coding, distributed inference and message passing algorithms.

Bhaskar Krishnamachari is currently an associate professor and Ming Hsieh Faculty Fellow at the USC Viterbi School of Engineering. He received his B.E. from the Cooper Union for the Advancement of Science and Art in 1998, and his M.S. and Ph.D. from Cornell University in 1999 and 2002 respectively, all in electrical engineering. He has co-authored more than 100 technical publications, mostly in the area of wireless networks, including papers that received best paper awards at IPSN 2004 and MSWiM 2006. He received the 2004 NSF CAREER Award and in 2005 received the outstanding junior faculty research award in engineering. He is an Editor for several journals: IEEE Transactions on Mobile Computing, ACM Transactions on Sensor Networks, ACM Mobile Computing and Communications Review, Elsevier's Ad Hoc Networks Journal, and Elsevier's Pervasive and Mobile Computing Journal. He has been TPC chair for EWSN 2010 and DCOSS 2009, and TPC Vice Chair for IEEE SECON 2009.

Fan Bai (General Motors Global R&D) is a Senior Researcher in the Electrical & Control Integration Lab., Research & Development and Planning, General Motors Corporation, since Sep., 2005. Before joining General Motors research lab, he received the B.S. degree in automation engineering from Tsinghua University, Beijing, China, in 1999, and the M.S.E.E. and Ph.D. degrees in electrical engineering, from University of Southern California, Los Angeles, in 2005.

His current research is focused on the discovery of fundamental principles and the analysis and design of protocols/systems for next-generation Vehicular Ad hoc Networks(VANET), for safety, telematics and infotainment applications. Dr. Bai has published about 50 book chapters, conference and journal papers, including Mobicom, INFOCOM, MobiHoc, SECON, ICC, Globecom, WCNC, JSAC, IEEE Transaction on Vehicular Technology, IEEE Wireless Communication Magazine, IEEE Communication Magazine and Elsevier AdHoc Networks Journal. He received Charles L. McCuen Special Achievement Award from General Motors Corporation "in recognition of extraordinary accomplishment in area of vehicle-to-vehicle communications for drive assistance & safety". He serves as Technical Program Co-Chairs for IEEE WiVec 2007, IEEE MoVeNet 2008, ACM VANET 2011 and ACM VANET 2012. He is an Associate Editor of IEEE Transaction on Vehicular Technology and IEEE Transaction on Mobile Computing, and he also serves as guest editors for IEEE Wireless Communication Magazine, IEEE Vehicular Technology Magazine and Elsevier AdHoc Networks Journal. He is also serving as a Ph.D. supervisory committee member at Carnegie Mellon University and University of Illinois - Urban Champaign.