Distributed Storage Codes Reduce Latency in Vehicular Networks [*]

Maheswaran Sathiamoorthy
University of Southern California
CA, USA
msathiam@usc.edu

Alexandros G. Dimakis
University of Southern California
CA, USA
adimakis@usc.edu

Bhaskar Krishnamachari
University of Southern California
CA, USA
bkrishna@usc.edu

Fan Bai
General Motors Global R&D,
ECI Lab,
Warren, MI, USA
fan.bai@gm.com

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 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 simple 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 number of replicas of each stored file. 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 two large sets of real vehicle traces from Beijing and 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.

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]. Applications that concern entertainment and file sharing are also receiving attention and involve different challenges (e.g., AdTorrent [4], FleaNet [5], CarTorrent [6], C2P2 [7]).

Content access and vehicle file sharing would enable users to access movies, music, real-time videos of distant road conditions, and other relevant content. In this paper we investigate the possibility of exploiting inter-vehicular communication to enable peer-to-peer file sharing with no infrastructure. One alternate possibility 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. Another option is to use infrastructure access points, but they may be hard to deploy in high densities (for example on freeways, access points may be separated by 5-10 miles [8]). In addition, due to the latency of content access from the Internet, a vehicle quickly passing through an infrastructure might not have sufficient time to download its desired data. In contrast, the WAVE mode of IEEE 802.11p allows for rapid vehicle to vehicle file transfers 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

[*]This research was sponsored in part by General Motors Research and Development, and by the U.S. Army Research Laboratory under the Network Science Collaborative Technology Alliance, Agreement Number W911NF-09-2-0053

network architecture which integrates vehicle to vehicle communication with vehicle to infra-structure communication.

The goal of this paper is to investigate how to store information in vehicles to optimize the peer-to-peer collaboration opportunities. 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 vehicle 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 simple memoryless process. We show how both replicated and encoded storage correspond to different balls and bins processes. Using simple stochastic dominance and coupling arguments, we bound the expected download time and obtain bounds for different parameter values. 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 $\mathcal{C}/m$. 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 benefit of N/α over replication, where N is the number of vehicles and α the number of replicas of each stored file. This means that, in this regime, coding yields the same performance as replicating all the files at all other vehicles, but using much less storage.

Beyond our analytical model, we present a practical performance analysis using real vehicle traces combined with a realistic 802.11p DSRC Packet Delivery Ratio (PDR) model. In particular, we use vehicle traces involving 1,000 taxis in Beijing, and 1,608 buses in Chicago. 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 of large files. For instance, we show that for downloading 1GB files, 98% of the files can be downloaded via vehicle-to-vehicle interactions in a day using coded storage, while only 19% of the files are downloaded in the same time period if uncoded replication is used.

The remainder of the paper is organized as follows: in the following section, we discuss the background and related work. In Section 3, we introduce the idea of distributed storage codes for vehicular networks. In Section 4, we detail the assumptions underlying our analytical model. In Section 5 we summarize the key analytical results showing when coding is beneficial in reducing latency of content access and in Section 6, we present the details of the analysis of latency with and without coding. Section 7 gives the details and results of our trace-based simulation experiments. In section 8, we briefly consider the challenging problem of optimal storage in the case of non-uniform file popularity. We conclude in Section 9. Appendix A is useful for the interested reader to understand stochastic dominance, which is used to get bounds on the expected delay in coded storage and, finally, appendix B gives some related detailed probabilistic results.

2. BACKGROUND AND RELATED WORK

In this paper we investigate how to store m files, each of size $\mathcal{M}$, in N vehicles. A sink vehicle that is interested in downloading one of the files is going to meet each of the storage vehicles according to a random process dictated by mobility. Every time the sink vehicle meets some other vehicle, it asks for chunks of the desired file and all nodes are assumed altruistic and helpful. One storage strategy is simple replication: each of the m files is stored in α nodes. 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 [9] 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.* [10] and later developed by Luby [11] and Shokrollahi [12], digital fountain codes are binary near-MDS (almost all sets of $k(1+\epsilon)$ chunks suffice to reconstruct the file with high probability) 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.* [13] and Kamra *et al.* [14]. Related comparisons of erasure coding versus replication are given for DHTs and DTNs, [15, 16, 17]. The use of coding in the form of mixing of packets in intermediate nodes for content distribution systems was first proposed in the context of a content delivery system called Avalanche [18, 19].

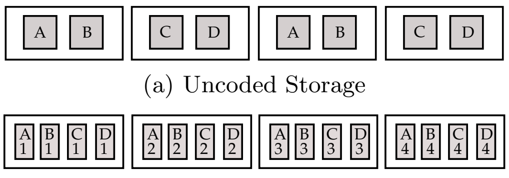
CarTorrent [6, 8, 20] is a peer-to-peer sharing architecture specifically tailored for VANETs which enables cooperative downloading of content. Also related is the C2P2 file sharing system [7] in which files are replicated, without coding, for low-latency access.

The use of network coding to disseminate content more efficiently in vehicular network is considered in CodeTorrent [21], VANETCODE [22], and VCD [23]. These works focus primarily on file distribution with in-network coding and do show the benefits of network coding. However, in contrast to these works on network coding in VANET, our work in this paper advocates a simpler pre-coded storage for on-demand file retrieval (which is not network coding *per se* because it does not involve any in-network recombination of packets). Moreover, in contrast to these prior works, we consider the problem mathematically and present analytical results on the benefits that can be obtained with coding, in addition to evaluation over real vehicular traces. Our work, for the first time, shows clearly the conditions under which coded storage provides latency benefits in a vehicular environment (namely large file sizes and low download bandwidth) and when it does not (small files and unconstrained bandwidth).

Finally, we note that our theoretical analysis relies on balls and bins processes (see e.g. [24, 25, 26]) and simple stochastic dominance arguments [27] that are used to obtain bounds on the expected delay for coded storage.

3. DISTRIBUTED STORAGE CODES

We first present the potential benefit of coded storage in vehicular networks with an illustrative example. Consider a system of four nodes each with same amount of storage capacity and four files A, B, C and D all having identical file sizes, such that the capacity of a node equals twice the file size. So it is possible to store each file twice throughout the nodes. There is a sink node that follows a simple memoryless contact process so that it meets one of the four nodes uniformly at random at each time step. Suppose the sink can only download half the file size every time it sees a node, and it is interested in downloading file A.



(b) Coded Storage

Figure 1: A simple example illustrating the benefits of coding.

Fig 1(a) shows one possible uncoded replication strategy for this example. The expected delay in downloading the file A is $2 + 2 = 4$, since the expected delay to see either the first or the third nodes is 2 each.

Now suppose a (n, k) coding is used before storing the files, with $n = 4$ and $k = 2$ in this case, so that each file is split into two chunks and then coded into four chunks. One example of such a coding would be to split A into A_1 and A_2 and then get $A_3 = A_1 + A_2$ and $A_4 = A_1 + 2A_2$, where each of the chunks can be represented as one element of a finite field $GF(3)$. If more than one finite field elements are required to represent one chunk, the same erasure code can be used multiple times. It can be easily seen that *any* two chunks out of the four can be used to reconstruct the file, since any two chunks correspond to two full-rank linear equations that can be solved for A_1, A_2. Such an allocation is shown in Fig 1(b). In this example, each chunk is half the file size and we assume that whenever a node is met by the sink, a chunk can be fully downloaded. When coding is used, the first time any node is seen, a useful chunk can be downloaded, and the second time a node is seen, it is *good* (i.e. it has a useful chunk) with probability $3/4$, and thus, the expected delay in this case is $1 + 4/3 = 2.33$. Clearly, in this example, the expected delay in the coded storage case is lower than the delay in the uncoded storage case.

4. 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 vehicles (or nodes) in a closed system of a vehicular network, 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. We require that $\mathcal{C} \geq \mathcal{M}$, so that a vehicle can store at least one file. Our theoretical analysis further requires that all files are equally popular (in section 8 we discuss how to deal with non-uniform file popularity).

It is desired to distribute these m files among the nodes. It is assumed that the total available storage exceeds the total size of files: i.e., that $N\mathcal{C} \geq m\mathcal{M}$. Denote $\alpha = \frac{N\mathcal{C}}{m\mathcal{M}}$; note that we can store each file $\alpha \geq 1$ times throughout the system and saturate the available capacity in the system. 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 two methods of storing files are used as described above - namely the uncoded storage scheme and the coded storage scheme. For the uncoded storage scheme, we simply store each file α times in the nodes ensuring deterministic 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 $N\mathcal{C} = mn\mathcal{M}/k$, yielding $\alpha = n/k$. Also note that if all the files are equally popular, the storage scheme as well as the expected delay for each file is going to be the same.

We focus on the latency experienced by a given sink vehicle node that is trying to download one of the m files. For the analysis, we assume a simple i.i.d. encounter model whereby 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**. 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 a realistic link quality model.

Given all other parameters, such as the file size, bandwidth constraint etc., we would like to determine the optimal values of n and k. In order to do so, we note that each chunk has size $\mathcal{M}/k$ and so we want to choose k such that this size is downloadable within the bandwidth constraint. Thus we want $\mathcal{M}/k \leq d$ or $k \geq \mathcal{M}/d$. But since higher k equates to higher coding complexity, we use $k = \lceil \mathcal{M}/d \rceil$. But if $d > \mathcal{M}$, since k cannot be smaller than 1, choose $k = 1$. Thus $k = \max(1, \mathcal{M}/d)$. The chunk size is therefore either $\mathcal{M}$ or d whichever is lower. Note that $k = 1$ in fact corresponds to not using any coding at all. One can choose a higher value of k, but we shall see later that it does not help to reduce the latency. 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. A $\beta > 1$ 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 and it is our objective in the analysis to quantify and minimize the expected latency $\mathbb{E}[D]$.

5. THE BENEFITS OF CODING

In this section we informally summarize our analytical results and explain why coding is beneficial in reducing latency in vehicular networks. Precise mathematical statements and proofs are given in the subsequent analysis section. Out first result is a simple derivation of the expected delay under uncoded replication (see equation 2 proved in section 6.1):

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

We also obtain an upper bound on the expected delay when coding is used (from equation 6, which will be derived in section 6.2):

$$\mathbb{E}[D_{\text{coded}}] \leq \max(N, n)\log\left(\frac{\alpha}{\alpha - 1}\right) \approx \frac{N}{\alpha}\max\left(1, \frac{\mathcal{M}}{d}\frac{\alpha}{N}\right)$$

where the approximation holds for large values of α.

It is immediately clear by comparing the above expressions, that the expected delay with coding is at least as good or better than without coding. By looking at the above expressions, we see that 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)* $d \geq \mathcal{M}$: In this case, a full file can be downloaded in one interaction and thus there is effectively no bandwidth limitation. Also, since $k = 1$, coding becomes equivalent to uncoded file replication. The expressions for the latencies in both the coded storage and uncoded storage schemes become almost equal in this case with $\mathbb{E}[D_{\text{uncoded}}] = N/\alpha \approx \mathbb{E}[D_{\text{coded}}]$, and so the improvement factor is approximately 1. Even if we make $k > 1$ to use coding, the probability with which the sink meets a useful node increases (because more nodes have the chunks), but the amount of data it can download decreases, hence higher values of k do not offer any benefit. A potential problem with coding with higher k in this case is that, if the sink itself contains a few files (different from the model used for analysis), then the delay to access the files the sink already contains is zero and thus uncoded storage scheme may offer a lower average delay; we will see this effect later in the numerical simulations.
- *(Intermediate Bandwidth regime)* $\mathcal{C}/m \leq d \leq \mathcal{M}$: The bandwidth constraint is smaller than the file size and thus when using an uncoded storage scheme, it is not possible for a sink to download the entire file when it meets a node. Thus it needs multiple encounters of nodes that contain the file the sink is interested in, before being able to download the entire file. In the coded storage scheme, the chunk size d is such that any node cannot store chunks of all the files since $dm \geq \mathcal{C}$. Thus the sink has to wait until it meets a useful node, which is the only factor contributing to the delay (as opposed to the bandwidth constraint). 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 factor is $\frac{\mathcal{M}}{d}$, whose value ranges from 1 to N/α as d decreases from $\mathcal{M}$ to $\mathcal{C}/m$.
- *(Bandwidth limited regime)* When $d \leq \mathcal{C}/m$, the sink is still severely bandwidth constrained under the uncoded storage scheme. For the coded storage scheme, each node can store a chunk of each file since $dm \leq \mathcal{C}$. And since each encounter gets the sink d amount of data, the sink should need about $\mathcal{M}/d$ encounters to successfully recover the file. Using the above

fact that $d \leq \mathcal{C}/m$ and $\alpha = N\mathcal{C}/m\mathcal{M}$, we obtain $\mathbb{E}[D_{\text{coded}}] \leq \frac{\mathcal{M}}{d}$. Since, $\mathbb{E}[D_{\text{uncoded}}] = \frac{N}{\alpha}\frac{\mathcal{M}}{d}$, the improvement factor here is at least N/α. Thus under such a severe bandwidth constraint, coding performs as if full files were available in all the nodes, only to be limited by the bandwidth.

6. THEORETICAL ANALYSIS

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 being downloaded 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.

6.1 Uncoded File Storage

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

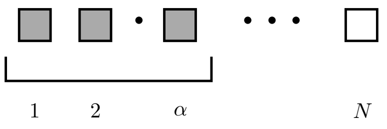


Figure 2: A vehicular network with N nodes and redundancy α represented in the balls and bins framework. Each node is represented as a square, with the shaded squares containing copies of the file the sink is interested in. Note that since a file is not stored twice in the same node, there are α nodes containing the copies of the files

In an uncoded storage scheme, 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 $\mathcal{C} >$ filesize $\mathcal{M}$, 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, $\mathbb{P}[\text{a node is } good] = \alpha/N$. Thus the number of nodes to be seen before encountering a good node is a geometric random variable with mean

$$\mathbb{E}[D] = \frac{N}{\alpha}. \tag{1}$$

Alternatively, in the balls and bins framework of Fig 2, this corresponds to throwing balls into N bins where each ball can land into any one bin with equal probability. We are interested in counting the average number of balls to be thrown before a ball lands into one of the shaded bins.

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 $\mathbb{E}[D] = \frac{N}{\alpha}\left(\frac{\mathcal{M}}{d}\right)$. Again, in the balls in bins framework, the equivalent formulation is to determine the number of balls in expectation that must be thrown till 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,

$$\mathbb{E}[D] = \frac{N}{\alpha}\max(1, \mathcal{M}/d). \tag{2}$$

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

6.2 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. We will address how to distribute the chunks to the nodes and the choice of k shortly. For a fixed value of the redundancy factor α, in the uncoded case, each file is replicated α times whereas in coded storage, each file is expanded, again, $\alpha = n/k$ times. Thus, analyzing the delays of both cases for the same α forms a fair comparison.

6.2.1 Balls and Bins model

Assume for a moment that β has to be an integer greater than or equal to 1 (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 of each bin to be β, meaning each bin can only fit β balls. See Fig 3. 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 $\lfloor\beta\rfloor$. 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 $\beta \leq$, $1 < \beta < k$ or $\beta \geq k$.

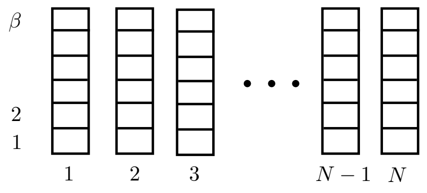


Figure 3: The balls and bins model for the coded case for integer $\beta \geq 1$

Case I: $\beta \geq k$

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

Case II: $\beta \leq 1$

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 nodes out of the N store the chunks. Without loss of generality, consider the first n nodes to contain the chunks. In the balls and bins process, assume that there N bins and we are interested in counting the number of balls in expectation to be thrown till there are k balls in any of the first n bins (see Fig 4). 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,

$$\begin{aligned}\mathbb{E}[D] &= \frac{N}{n} + \frac{N}{n-1} + \frac{N}{n-2} + \ldots + \frac{N}{n-k+1}\\ &= N(H_n - H_{n-k})\\ &\approx N \log\left(\frac{n}{n-k}\right),\end{aligned}$$

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

$$\mathbb{E}[D] \approx N \log\left(\frac{\alpha}{\alpha - 1}\right). \tag{3}$$

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.

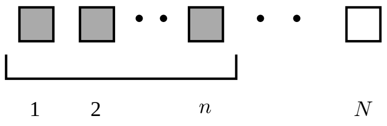


Figure 4: The Bins set up when $\beta < 1$

Case III: $1 < \beta < k$

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, otherwise some bins will have capacity $\lceil\beta\rceil$ and remaining will have capacity $\lfloor\beta\rfloor$, making bins non-identical and thus harder to analyze. 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

$$\mathbb{E}[D] = \sum_{i=0}^{k-1} \mathbb{E}[T_{i\to i+1}]. \tag{4}$$

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), then $T_{0\to 1}$ 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 state 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 $\text{Geom}(q)$ is a geometric random variable with failure probability q. We make a note that $T_{\beta\to(\beta+1)}$ is a probabilistic mixture of two geometric random variables (refer to Appendix A for the definition of probabilistic mixtures). 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 it, for which we need to introduce some of the techniques below (also see [27, 29] 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 $z \in R$, the following probability inequality must hold

$$\mathbb{P}(X \leq z) \geq \mathbb{P}(Y \leq z)$$

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.

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

Lemma 6.1. *Let $X \sim$ Geom(p) and $Y \sim$ Geom(q), where p and q are the failure probabilities. If $p \leq q$, then $X \preceq Y$.*

Lemma 6.2. *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 $X \preceq X_1$.*

Back to the case when $i = \beta + 1$, since the worst case failure probability is $1/N$, the corresponding geometric random variable stochastically dominates other geometric random variables (Lemma 6.1), and so from Lemma 6.2, 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 worst case 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 6.3. *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 overflown 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 worst case 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 bins full, and so in all these cases, the worst case 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 worst case 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 6.1 and 6.2. This implies $\mathbb{E}[T_{i \to i+1}] \leq \mathbb{E}[\text{Geom}(p'_i)]$ and thus $\mathbb{E}[D] \leq \mathbb{E}[D']$.

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

Noting that $\mathbb{E}[\text{Geom}(p)] = \frac{1}{1-p}$,

$$\begin{aligned}
\mathbb{E}[D'] &= \sum_{j=0}^{x-1} \sum_{i=j\beta}^{(j+1)\beta-1} \frac{1}{1 - j/N} = \sum_{j=0}^{x-1} \frac{\beta}{1 - j/N} \\
&= N\beta(H_N - H_{N-x}) \\
&= n(H_N - H_{\lceil N(1-1/\alpha) \rceil}).
\end{aligned}$$

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

$$k \leq \mathbb{E}[D] \leq n(H_N - H_{\lceil N(1-1/\alpha) \rceil}). \tag{5}$$

□

The right hand side can also be approximated as $k\alpha \log(\frac{\alpha}{\alpha-1})$. When using this expression, it does not matter whether x or β is an integer or not. As $\alpha \to \infty$, we can further approximate the above equation as $k \leq \mathbb{E}[D] \leq k[1 + 1/2\alpha + o(1/\alpha^2)]$. Thus $E\{D\} = k$ for large α. Another way to see this is to note that $n = N\beta$ denotes the total capacity of all the bins and α measures how big the capacity is compared to k. Therefore as $\alpha \to \infty$, k is much smaller than the capacity and thus none of the bins would overflow and so the average number of balls required is approximately k. Also note that the probability that the delay is greater than

k decreases exponentially with α^2. The details are in Appendix B. This is useful because we gain a lot even if the redundancy increases a little bit.

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,

$$\mathbb{E}[D] \leq \max(1, \beta) N \log\left(\frac{\alpha}{\alpha - 1}\right). \quad (6)$$

The $\beta \geq k$ is not interesting because it implies $\beta = n/N \geq k$ or $\alpha = n/k \geq N$. In practice, this will never be reached and if it does, then because N will generally be high, from equation 5, we get $\mathbb{E}[D] \approx k$.

Comparisons of the expected delay with uncoded and coded storage based on this analytical framework are shown in Fig 5. Not only is the benefit of coding is apparent from this figure, we can also see that the simulation matches quite well with the analysis.

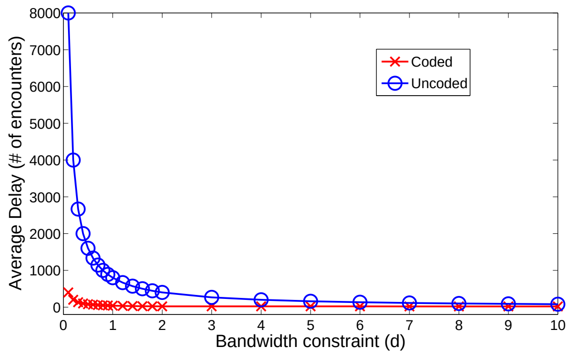


Figure 5: A plot comparing the expected delay when using coded storage vs uncoded storage under the analytical framework. Here, number of nodes N=400, storage per node $\mathcal{C}$ = 50MB, number of files $m = 25$ and file size $\mathcal{M}$ = 40MB.

A note on the choice of k: All the expressions obtained above for the expected delay correspond to the case when k chosen such that the chunk size $\mathcal{M}/k$ is equal to the bandwidth constraint 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 say twice its actual value, 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.

7. TRACE BASED EXPERIMENTS

We now turn to an empirical evaluation of the benefits of coded storage, using real vehicular traces. For the real traces of vehicles, we use two datasets: one consisting of GPS traces of taxis in Beijing, and another consisting of traces of buses in Chicago. The Beijing dataset consists of coordinates of 1,000 taxis in Beijing from 00:00hrs till 23:59hrs on Jan 5, 2009 local time, recorded approximately every minute. For the Chicago dataset, we collected the GPS traces of 1,608 buses in Chicago, using the API released by the Chicago Transit Authority(CTA)[1]. We started collecting this data from 11:06hrs (Chicago local time) on Nov 1, 2010, recording the coordinates of buses about every 30 seconds, and used data of the first 24 hours. We primarily present results from the Beijing dataset, but we do include some results using the Chicago dataset, which shows similar trends, while omitting others due to lack of space. We assume that the nodes continue to run their application throughout the day. For inter-vehicular communication, we used a realistic model of IEEE 802.11p from [30], the details are given in section 7.2 below.

In order to characterize the performance of the system on these real traces, we cannot simply use the average delay in downloading a file as a figure of merit, like we did before for the analysis. 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 incomplete files. Thus, for our experiments 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. Certain files may be unusable if they are not fully reconstructed, and in such cases, this metric will be useful. There could be other cases, where it is enough to just measure the file reconstruction percentage by a given time. In such cases, we use *average file download percentage*, which as the name indicates, measures, on average, how much of a file is downloaded by a given time. The exact details on how these metrics are measured are given in section 7.3 below. Before proceeding further, we need to emphasize the difference between the two metrics and why the average file download percentage is insufficient in itself. Consider as an example, a situation where the average file download percentage is 60. This could mean that all the sinks have downloaded only 60% of the file each is interested in or 60% of sinks have been able to download full files but the remaining sinks nothing. Thus, it is also essential to measure how many sinks have been able to download complete files.

7.1 Simulation Setup

Similar to our analysis, both in the uncoded and the coded storage schemes, the chunks and files are stored

[1]http://www.transitchicago.com/developers/bustracker.aspx

by ensuring maximal spreading, so that, in the case of uncoded storage, a file is not stored in the same node twice 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, instead of performing maximal spreading, if the files or chunks are stored by selecting nodes randomly, we found that 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.

The next step after data storage is to simulate the movement of nodes as dictated by the GPS traces of the datasets. Since the traces are one day long, we divide the day into discrete time steps that are one minute long for the Beijing dataset and 30 seconds long for the Chicago dataset. Then, at each time step, all we need to do is to 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.

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 in a figure. By default, files are assumed to be of size 1GB, typical of movie clips. The number of files that can be distributed depends on the number of nodes and thus the total available capacity, thus we consider a default of 2,500 files in the system, so that each file can be replicated $\alpha = 40$ times when there are 1,000 nodes.

7.2 Realistic Radio Link Model

The IEEE 802.11p standard specifies the data rate to be from 1.5Mbps till 27Mbps with the default being 3Mbps. Hence, in our simulations, we use 3Mbps as the data rate. For inter-vehicular communication, we use an empirical model of packet delivery characteristics obtained from [30]. The authors in [30] 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 datasets is the Suburban Road (SR) enviroment. Thus we use the PDR vs separation distance data (Fig 3(a) in [30]) to carry out our simulations. It may also be emphasized that the authors found that the relative velocity between two nodes do 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.

7.3 Experimental Methodology

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. Once the files or the chunks (depending on the scheme to evaluate) are stored in all the nodes, in order to simulate the file sharing application, a random node is selected to be the sink and it tries to collect a particular random file. For each sink-file pair, we keep track of the percentage of the file completed and whether the file is completed or not at each time step. When presenting the simulation results, we average over 50 random sinks, each of which collects 100 files at random, to obtain the average file completion ratio and the full-recovery probability.

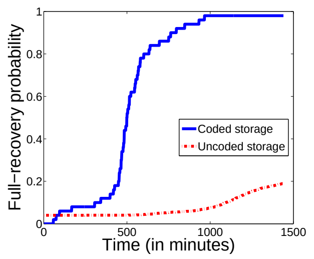
It may be noted that these are the average file completion curves, and that any curve ending at lesser than 100% file completion does not mean that the files were never fully reconstructed, it just means that there were a few files which were not reconstructed fully.

7.4 Choice of the coding parameter k

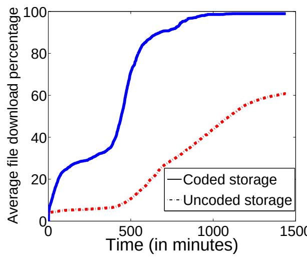
From the analysis, we deduced that the choice of parameter k must be such that when a file of size $\mathcal{M}$ is split into k chunks of each size $\mathcal{M}/k$, this chunk size should be lesser than or equal to the bandwidth constraint d, to effectively overcome the bandwidth limitation. But, in practice, there is no fixed bandwidth constraint as such, since the amount of data that can be downloaded depends on the contact duration (among other factors), and hence it is not possible to use the above equation to calculate the optimal value of k. From the Beijing dataset, we noticed that the average contact duration between two nodes is about 55.6s (the duration when two nodes are within a distance of 500m), and hence at 3Mbps datarate, the upper bound on the average amount of data that can be transferred is 21MB (corresponding to the PDR being 1 throughout the contact duration, which will hardly be the case and hence this quantity will be lower). Thus a reasonable choice of the chunk size would be around 1MB or lower. In fact, we noticed that the improvement (in say the average file completion ratio) was trivial when choosing 0.1MB as the chunk size instead of 1MB and hence in all our simulations, k is chosen so that each file gets split into 1MB chunks. For example, for a 1GB file, we would need k=1000 (considering 1GB=1000MB).

7.5 Discussion of the Results

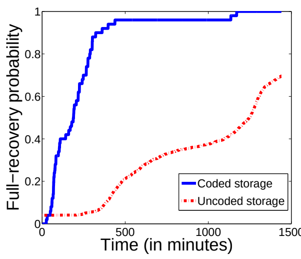
Our most important results are shown in Fig 6, in which we consider a typical file sharing scenario with 2,500 files each of size 1GB; and each node having about 100GB storage. Such a system is implemented atop both the Beijing and the Chicago datasets, and both the full-recovery probability and the average file down-



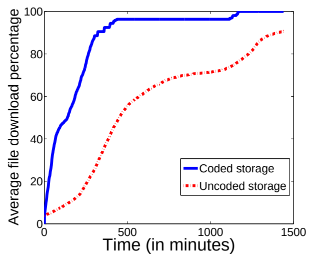
(a) Beijing dataset (full-recovery probability)



(b) Beijing dataset (average file download percentage)



(c) Chicago dataset (full-recovery probability)



(d) Chicago dataset (average file download percentage)

Figure 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. The number of nodes is 1,000 in the Beijing dataset and 1,608 in the Chicago dataset

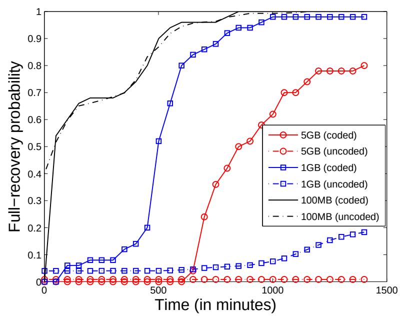
load percentage are measured for each time step. The plots are shown in Fig 6. We note that coding offers significant benefits compared to no coding. For example, at the end of 24 hours, by using coding, as much as 98% of the time, files are reconstructed completely, whereas without coding, only 19% of the files are reconstructed fully (see Fig 6(a)). On the other hand, if we were to only consider the percentage of file completion, then sinks are able to download 99% of the files on average when coding is used, whereas without coding, only 61% of the files are downloaded. 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 Beijing trace when coding is used, but only 4.4% of nodes are successful in full downloads by 600 minutes if no coding is used.

For the Chicago dataset (Fig 6(c) and Fig 6(d)), the full-recovery probability at the end of 24 hours is 1 when coding is used, whereas when no coding is used, the probability is 0.7. The average file download percentage for the coded and uncoded cases at the end of 24 hours turned out to be 100% and 91% respectively. After about 287 minutes, 80% of sinks succeed in getting full files when coding is used, whereas if no coding is used, only 6% of the nodes were able to complete their downloads by that time. We see that the general trends in the simulations from both traces are similar, though the Chicago trace yields lower latencies. The improvements in latency observed in the Chicago dataset could be partially attributable to the higher redundancy factor because it involves a larger number of cars (1,608 instead of 1,000), but also to other factors such as the fact that the Beijing trace begins in the middle of the night with relatively little traffic, while the Chicago trace begins late in the morning. One can see from Fig 6(a) 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.). In contrast, the rate at which files are completed is initially steady for the Chicago dataset, since it begins around 11 a.m., but is almost flat during the night and then resumes again next day morning. Another factor affecting the rate towards the end is the scarcity of new chunks (similar to the coupon collector problem).

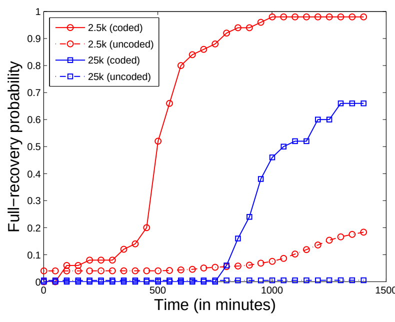
Further, we performed a number of experiments to thoroughly understand the effect of various parameters on the performance of the system, by systematically varying the parameters. There are many parameters that can be varied, namely, the number of nodes N; the storage per node $\mathcal{C}$; the number of files m; the size of each file $\mathcal{M}$; the system redundancy α; and the coding parameters (n, k). It may however be noted that these parameters are all interconnected by the relationship $\alpha = N\mathcal{C}/m\mathcal{M} = n/k$. k is fixed once $\mathcal{M}$ is known (as explained above), and once we know α, n can also be fixed. The number of nodes N is determined by the dataset; it is 1,000 for the Beijing dataset and 1,608 for the Chicago dataset. Thus we are left with three free parameters $\mathcal{M}$, $\mathcal{C}$ and m and to study their effect, in our evaluations, we keep two parameters constant and vary the third parameter.

7.5.1 Effect of file size

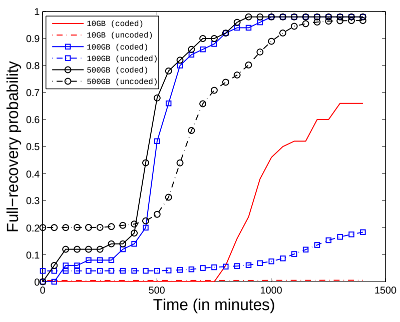
As file sizes increase, since system storage remains constant, we are effectively decreasing the system redundancy, which should adversely impact latency. This is observed for both coded and uncoded storage, but there are clear differences in relative performance. From the analysis, we inferred that if the bandwidth constraint d is about the same or higher than the file size $\mathcal{M}$, then coding offers little to no benefit. We notice this effect in our simulations when the file size is about 100MB or lower. This is shown in Fig 7(a) and Fig 8(a). But we can see that, as the file size is increased to 1GB, coding offers tremendous improvements by being able to fully download the files most of the time (98% of the time), whereas only about a fifth of the time that sinks are able to download full files by the end of the



(a) File size varied keeping other parameters constant

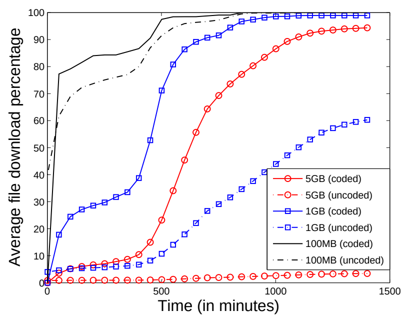


(b) Number of files varied keeping other parameters constant

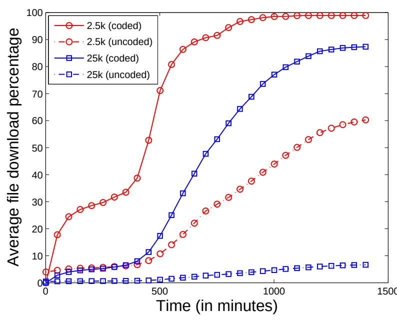


(c) Capacity per node varied keeping other parameters constant

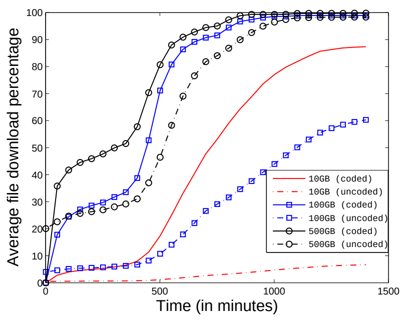
Figure 7: Plots showing how various parameters affect the full-recovery probability. In each of the cases, one parameter is varied while keeping the others constant. Typical values used are a storage capacity of 100GB, 2,500 files and file size 1GB. Beijing dataset is used for all the simulations.



(a) File size varied



(b) Number of files varied



(c) Capacity per node varied

Figure 8: Plots showing the impact of different parameters on the average file download percentage. The parameters are same as in Fig 7 and the simulations are carried out on the Beijing dataset.

simulation. 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 8(a), we note that many sinks have been able to download about a tenth of the file on the average, but not a complete file).

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

Figs 7(b) and 8(b) 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 the the full-recovery probabilities and the file download percentages both start to get better (Fig 7(c) and Fig 8(c)). 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 7(b) and the curve corresponding to 2,500 files with 10GB storage per car in Fig 7(c) (or Fig 8(b) and Fig 8(c)) 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.

7.6 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 more than 1000 nodes each, are still relatively quite sparse in terms of encounters as they involve two very large cities (Beijing, Chicago). Further, it is important to note that the Beijing simulations start around midnight, which also skews the latencies observed in that data set as there is not much encounter activity till many hours later (as noted before, the latencies are lower in the Chicago trace which starts from late-morning). 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 as well as in our analytical model indicate strongly that the use of coding is essential for speeding up large file downloads in encounter-based vehicular networks, regardless of vehicular density.

8. NON-UNIFORM FILE POPULARITY

In both the analysis and the preceding trace-based simulations, we have assumed that all files are equally popular. We now relax this assumption and consider non-uniform file popularity. In the uncoded storage case, if each file has a popularity p_i such that $\sum_{i=1}^{m} p_i = 1$, and if all encounters are equally likely, then we know that the latency of accessing each file is inversely proportional to the number of replicas (and the bounds we derived for coded storage case also show a similar inverse dependency). In such a setting, it is known that file i must be replicated ideally as per the square-root allocation strategy [7, 31] in order to minimize the average latency over all files. In other words, the number of copies of file i should be α_i, where:

$$\alpha_i = m\alpha \frac{\sqrt{p_i}}{\sum_{j=1}^{m}\sqrt{p_j}}. \qquad (7)$$

In the real traces, where encounters are non-uniform because of the arbitrary mobility pattern, the assumption that the latency is strictly inversely proportional to the redundancy factor may not hold, either for uncoded or for coded storage. Determining the optimal storage policy is thus challenging and we defer it to future work. Here, however, we consider the square-root replication policy as a reasonable sub-optimal heuristic and evaluate briefly the benefits it offers to both uncoded and coded storage settings in the case that the file popularity is non-uniform.

Fig 9 demonstrates the improvements that can be achieved by making use of square-root allocation strategy when compared to uniform allocation. In this case, it is assumed that the file popularities follow a Zipf law with parameter 0.77 (content requests on the web are known to follow such a Zipf-like distribution [32]). In both cases we see improvements, though they are more marked in the case of uncoded storage. But even in this case, we see that coded storage does perform better on

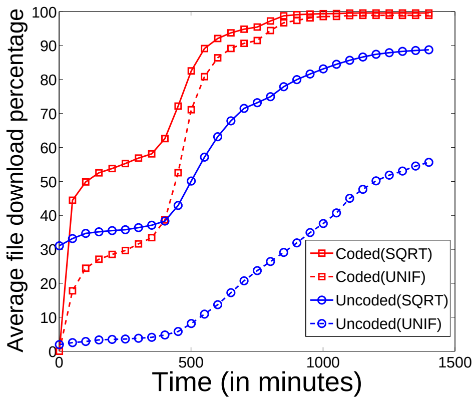


Figure 9: Comparing square root replication vs uniform replication for both the coded and uncoded storage schemes. The average file download percentage is plotted here for the Beijing dataset.

the whole.

9. CONCLUSION

We have studied the effect of coded storage on the latency of content access in an encounter-based 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 case of larger files and bandwidth limitations. We have further validated our findings using realistic simulations based on large-scale vehicular traces involving taxis in Beijing, and buses in Chicago. Our numerical results confirm that file download latency (particularly for larger files) is improved dramatically when the content is stored using MDS erasure codes.

There are still many unanswered questions. While we have briefly addressed the issue of unequal file popularity, a more rigorous solution remains to be developed. Other open questions include how to re-distribute content when there are nodes leaving and entering the system, and the possibility of learning patterns in vehicular encounters to further optimize the content storage. We also note that our traces, albeit involving 1000+ cars, are still relatively sparse given that they involve city-scale mobility; it is important to understand content access latency in denser deployments, to determine if vehicular P2P file sharing can become reality.

APPENDIX

A. COUPLING AND STOCHASTIC DOMINANCE

Defn: Probability Mixtures A probability mixture is a convex combination of a given set of probability distributions. Suppose there are a set of l discrete random variables $X_1, X_2, \dots, X_l$ whose probability mass functions are given by $p_{X_j}(\cdot)$ for $j = 1, 2, \dots, l$ and given mixing constants $\alpha_j \geq 0$ $(j = 1, 2, \dots, l)$ with $\sum_{j=1}^{l} \alpha_j = 1$, then a random variable Y formed out of a probability mixture of $X_1, X_2, \dots, X_l$ has the probability mass function:

$$p_Y(z) = \sum_{j=1}^{l} \alpha_j p_{X_j}(z).$$

In case of continuous random variables, the probability mass function will have to be replaced by the probability density function. It is not hard to see that the above equality also holds for cumulative distributions. $F_Y(z) = \sum_{y \leq z} p_Y(y) = \sum_{j=1}^{l} \alpha_j \sum_{y \leq z} p_j(y) = \sum_{j=1}^{l} \alpha_j F_{X_j}(z)$.

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

$$\mathbb{P}(\hat{X} \in E) = \mathbb{P}(X \in E)$$

THEOREM A.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*

$$\mathbb{P}(\hat{X} \leq \hat{Y}) = 1$$

PROOF. If $\mathbb{P}(\hat{X} \leq \hat{Y}) = 1$, then it means that $\hat{X} \leq \hat{Y}$ is always true and so $\mathbb{P}(\hat{X} \leq \hat{Y} | \hat{Y} \leq z) = 1$. Thus we have $\mathbb{P}\left[(\hat{X} \leq \hat{Y}) \cap (\hat{Y} \leq z)\right] / \mathbb{P}(\hat{Y} \leq z) = \mathbb{P}(\hat{X} \leq \hat{Y} \leq z) / \mathbb{P}(\hat{Y} \leq z) = 1$. Now,

$$\begin{aligned}\mathbb{P}(Y \leq z) &= \mathbb{P}(\hat{Y} \leq z) \\ &= \mathbb{P}(\hat{X} \leq \hat{Y} \leq z) \\ &\leq \mathbb{P}(\hat{X} \leq z) \\ &= \mathbb{P}(X \leq z)\end{aligned}$$

So $X \preceq Y$. In order to prove the other direction we make a note that $X = F_X^{-1}(U)$ where U is a uniform random variable in $[0, 1]$ and $F_X^{-1}(\cdot)$ is the inverse of the cumulative distribution function defined as below:

$$F_X^{-1}(u) = \inf\{x \in \mathbb{R} : F_X(x) \geq u\}$$

By the property of stochastic dominance, we have $F_X(z) \geq F_Y(z)$ which implies that for any $0 \leq u \leq 1$, $F_X^{-1}(u) \leq F_Y^{-1}(u)$. And since X and $\hat{X}$ and Y and $\hat{Y}$ have the same distributions, $\mathbb{P}(\hat{X} \leq \hat{Y}) = \mathbb{P}(F_X^{-1}(U) \leq F_Y^{-1}(U)) = 1$ □

LEMMA A.2. *Let $X \sim Geom(p)$ and $Y \sim Geom(q)$ 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 $(P)(\hat{X} \leq \hat{Y}) = 1$ giving $X \preceq Y$. Thus a geometric random variable is always dominated by another geometric random variable with higher failure probability. □

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

PROOF.

$$\begin{aligned}F_X(z) &= \sum_{j=1}^{l} \alpha_j F_{X_j}(z) \\ &\geq \sum_{j=1}^{l} \alpha_j F_{X_1}(z) \\ &= F_{X_1}(z)\end{aligned}$$

Thus we have $F_X(z) \geq F_{X_1}(z)$ for all z, which implies $X \preceq X_1$. □

B. PROBABILITY ANALYSIS

Let us find out the conditions under which, with high probability, no bin gets full in k throws so that the number of throws is k with high probability. Without any capacity constraints on the bins, we know that the number of balls that land on a particular bin after k throws is binomially distributed with parameters k and $1/N$. Define, E_i to be the event that the bin i doesn't get full after k throws. So $Pr[E_i] = Pr[X \leq \beta]$ where X is a binomial random variable with parameters k and $1/N$. Probability that no bin gets full in k throws: $p = \mathbb{P}[E_1 \cap E_2 \cap \dots E_N] = 1 - \mathbb{P}[\bar{E}_1 \cup \bar{E}_2 \cup \dots \bar{E}_N]$. By using union bound and noting that all bins are identical,

$$\mathbb{P}[\bar{E}_1 \cup \bar{E}_2 \cup \dots \bar{E}_N] \leq \sum_{i=1}^{N} \mathbb{P}[\bar{E}_i] = NPr[\bar{E}_1] \quad (8)$$

$$\text{So} \qquad p \geq 1 - N\mathbb{P}[\bar{E}_i] \quad (9)$$

The task now is to find an upper bound on the bad event, $\mathbb{P}[\bar{E}_i]$ which is the probability that a bin gets full for which we make use of Chernoff bound for binomial random variables which states that if X is a binomial random variable with parameters n and p, then

$$\mathbb{P}[X \geq (1+\epsilon)np] \leq e^{-\epsilon^2 np/3}$$

So in our case, $\mathbb{P}[\bar{E}_i] = \mathbb{P}[X \geq \beta] = \mathbb{P}[X \geq \frac{n}{N}] = \mathbb{P}[X \geq \alpha\frac{k}{N}] \leq e^{-\frac{1}{3}(\alpha-1)^2\frac{k}{N}}$. Substituting in equation (9), $p \geq 1 - Ne^{-\frac{1}{3}(\alpha-1)^2\frac{k}{N}}$. Further, if we want $p \geq 1 - 1/N$, then we need to make sure that the bad event probability is upper bounded by $1/N^2$. Thus,

$$e^{-\frac{1}{3}(\alpha-1)^2\frac{k}{N}} \leq \frac{1}{N^2}$$

$$\alpha \geq 1 + \sqrt{\frac{2N\log N}{k}}$$

Thus when the above condition holds, the probability that the delay is equal to k is greater than $1 - 1/N$.

C. ADDITIONAL AUTHORS

D. REFERENCES

[1] D. Jiang and L. Delgrossi, "IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments," in *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE.* IEEE, 2008, pp. 2036–2040.

[2] L. Armstrong, "Classes of Applications," *Presentation, http://www.leearmstrong.com/DSRC Home/General Info/Applications/Application Examples Overview.ppt.*

[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 *Proceeding of the 6th international conference on Mobile systems, applications, and services.* ACM, 2008, pp. 29–39.

[4] A. Nandan, S. Tewari, S. Das, M. Gerla, and L. Kleinrock, "Adtorrent: Delivering location cognizant advertisements to car networks," in *Proc. Third IEEE/IFIP Annual Conference on Wireless On-demand Network Systems and Services (WONS06).* Citeseer, 2006.

[5] 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.

[6] 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 *2007 Mobile Networking for Vehicular Environments*, May 2007, pp. 109–114.

[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] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Sanadidi, "Co-operative downloading in vehicular ad-hoc wireless networks," in *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, 2005, pp. 32–41.

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

[10] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication.* ACM, 1998, pp. 56–67.

[11] M. Luby, "LT codes," *Proc. IEEE Foundations of Computer Science (FOCS)*, 2002.

[12] A. Shokrollahi, "Raptor codes," *IEEE Trans. on Information Theory*, June 2006.

[13] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 111–117.

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

[15] R. Rodrigues and B. Liskov, "High availability in DHTs: Erasure coding vs. replication," in *Proc. IPTPS*, 2005.

[16] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: a quantitiative comparison," in *Proc. IPTPS*, 2002.

[17] 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.

[18] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4. IEEE, 2005, pp. 2235–2245.

[19] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive view of a live network coding P2P system," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement.* ACM, 2006, pp. 177–188.

[20] S. Das, A. Nandan, and G. Pau, "SPAWN: a swarming protocol for vehicular ad-hoc wireless networks," in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks.* ACM, 2004, pp. 93–94.

[21] U. Lee, J. Park, J. Yeh, G. Pau, and M. Gerla, "Code torrent: content distribution using network coding in vanet," in *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking.* ACM, 2006, pp. 1–5.

[22] S. Ahmed and S. Kanhere, "VANETCODE: network coding to enhance cooperative downloading in vehicular ad-hoc networks," in *Proceedings of the 2006 international conference on Wireless communications and mobile computing.* ACM, 2006, pp. 527–532.

[23] 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 *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 23:1–23:12. [Online]. Available: http://doi.acm.org/10.1145/1921168.1921199

[24] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations (extended abstract)," in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, ser. STOC '94. New York, NY, USA: ACM, 1994, pp. 593–602. [Online]. Available: http://doi.acm.org/10.1145/195058.195412

[25] J. Byers, J. Considine, and M. Mitzenmacher, "Geometric generalizations of the power of two choices," in *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures.* ACM, 2004, pp. 54–63.

[26] D. J. Newman and L. Shepp, "The Double Dixie Cup Problem," *American Mathematical Monthly*, vol. 67, no. 1, pp. 58–61, 1960.

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

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

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

[30] F. Bai, D. Stancil, and H. Krishnan, "Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular

network engineers," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking.* ACM, 2010, pp. 329–340.

[31] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM, 2002, pp. 177–190.

[32] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2002, pp. 126–134.