Collaboratively Replicating Encoded Content on RSUs to Enhance Video Services for Vehicles

Yipeng Zhou, Jiawen Chen, Guoqiao Ye, Di Wu, Jessie Hui Wang and Min Chen.

Abstract—With the development of smart cities, Internet services will be pervasively accessible for moving vehicles. It is envisioned that the video content demand of vehicles will explode in the near future. However, the strategy to efficiently distribute video content in large-scale vehicular networks is still absent due to challenges arising from the huge video population, heavy bandwidth consumption, heterogeneous user devices and vehicles' mobility. In this work, we propose to collaboratively replicate video content on Roadside Units (RSUs) to enhance video distribution services based on the fact that the contact period between moving vehicles and a single RSU is not long enough to complete video downloading. In our design, a video file is split into multiple chunks. Each RSU replicates a small number of original chunks and chunks encoded by network coding. Replicating encoded chunks can reduce redundancy of chunks on different RSUs so that RSUs can complement each other better, whereas original chunks can be transrated to chunks with lower bitrates flexibly to fit in users' devices. Therefore, we replicate both original and encoded chunks on RSUs to take advantages of both sides. Stochastic models are employed to analyze chunk download processes and a convex optimization problem is formulated to determine the optimal partition of space allocated to each kind of chunks. Furthermore, we extend our strategy to support video streaming services and empirically prove that the influence caused by limitations of network coding is moderate. In the end, we conduct extensive simulations which not only validate the accuracy of our models but also demonstrate that our strategy can effectively boost video distribution services.

Index Terms—Roadside unit, video file downloading, encoded chunks, vehicular networks

1 Introduction

Video distribution is one of the most crucial applications for modern Internet. In the future as vehicular networks are ubiquitously available, video services for moving vehicles[1] will explosively surge, which can trigger a variety of promising applications. We enumerate several typical examples here. Users can enjoy video contents [1], [2], participate video conferences/chats or download videos for future entertainment during their trips [3], [4]. One can broadcast advertisement in video form to nearby vehicles [5]. Users can share videos recorded by their vehicles with their friends or other vehicles [6], [7] in real time.

However, efficiently distributing video content for a large number of moving vehicles is a challenging problem. For vehicular networks, a tremendous number of Roadside Units (RSUs) will be deployed to provide Internet services [8], [9]. To enhance video distribution efficiency, a widely adopted approach is to replicate video content on RSUs by perceiving RSUs as mini-servers that are close to end users [3], [10]. A critical question arising here is which video files should be replicated on RSUs given limited storage space but a relatively infinite number of videos.

We tackle this problem from a new perspective by reducing the storage space occupied by individual videos so that more videos can be replicated on RSUs. As vehicles move, distributing large video files cannot be accomplished by a solo RSU since the contact period between vehicles and an RSU is very short. We leverage this property by only replicating a small fraction of content of each video on each RSU. However, without replicating complete video files, there is a risk that RSUs fail to provide complete videos for vehicles. Thus, RSUs should replicate complementary contents so that they can collaboratively serve moving vehicles.

Specifically, we split a large video file into multiple chunks and each RSU only replicates a small fraction of all chunks. The question is which chunks should be replicated on each RSU in order to maximize video distribution capacity. Through rigorous analysis, we prove that both original chunks and chunks encoded by network coding should be replicated. Intuitively speaking, we should replicate complementary chunks on different RSUs so that they can help each other. However, unless vehicles access RSUs with a fixed sequence, we cannot guarantee that chunks on different RSUs can complement each other. Fortunately, by encoding video chunks with network coding in advance, we can ensure almost all encoded chunks are independent and

This work was supported by ARC DE180100950, the National Natural Science Foundation of China under Grant U1911201, Guangdong Special Support Program under Grant 2017TX04X148, the Fundamental Research Funds for the Central Universities under Grant 19LGZD37, 19LGYJS57, 19LGYJS58.

Corresponding author: Di Wu.

Yipeng Zhou is with the Department of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, Australia (email: yipeng.zhou@mq.edu.au);

Jiawen Chen, Guoqiao Ye and Di Wu are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China and Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, 510006, China (emails: chenjw275, yegq3@mail2.sysu.edu.cn, wudi27@mail.sysu.edu.cn);

J. H. Wang is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology, Beijing 100084, China (e-mail: jessiewang@tsinghua.edu.cn);

Min Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, China (email: minchen2012@hust.edu.cn).

1. Vehicles are used interchangeably with users in this paper.

hence useful for users [11], [12], [13], [14]. [2] Random linear codes (as a kind of network coding techniques) is suitable for this scenario which can almost optimally generate an infinite number of encoded chunks [15]. As long as a certain number of independent encoded chunks are collected, the video can be recovered by decoding. Nevertheless, encoded chunks cannot be transrated to versions with lower bitrates to fit in heterogeneous user devices.[3] In other words, network coding isolates a video's different versions and we have to treat them separately. In contrast, an original chunk can be transrated to versions with lower bitrates flexibly. Thus, to achieve the best performance, it is necessary to suppress the weakness caused by purely replicating original or encoded chunks, and both kinds of chunks should be replicated on RSUs.[4]

We analyze video downloading processes with stochastic models and formulate a convex optimization problem to determine the optimal partition between the space for replicating each kind of chunks. We further extend our strategy to support video streaming services through sequentially downloading multiple video segments. We empirically prove that the influence caused by the limitations of network coding is moderate and our strategy is applicable in practical systems. Finally, extensive simulations are conducted to verify theoretical models and demonstrate the efficacy of our strategy in video distribution services.

The rest content of the paper is organized as follows. The state-of-the-art related works are discussed in Sec. 2. We introduce background and preliminary knowledge in Sec. 3. Stochastic models are presented in Sec. 4, following which the algorithm to partition replication space is proposed in Sec. 5. The case with heterogeneous RSUs is discussed in Sec. 6. In Sec. 7, we extend our strategy to support video streaming and discuss the overhead caused by network coding. At last, simulation results are elaborated in Sec. 8 before we conclude our paper in Sec. 9.

2 Related Work

Recently, how to efficiently and reliably distribute content especially video content in vehicular networks has attracted intense research study. We briefly introduce these works from three aspects: communication protocols, video content replication and network coding techniques.

2.1 Communication in Vehicular Network

Most of existing works focused on improving transmission efficiency in vehicular networks through enhancing collaborations between vehicles by forming ad hoc networks. In comparison with our work, they ignore the collaboration opportunities between RSUs. The works [4], [16], [17], [18] studied the problem to distribute content to vehicles via RSUs in ad hoc vehicular networks. They designed various vehicle-to-vehicle content sharing strategies so as to improve the content distribution performance. [6], [19], [20] investigated the scenario that vehicles need to upload their video contents such as surveillance videos to remote servers or other vehicles in an ad hoc manner in which vehicles help each other with transmission.

Some other works contributed to reduce collision possibilities when multiple vehicles need to transmit messages simultaneously. For example, [21] designed a detailed protocol for vehicle communications by adopting coding techniques to reduce collision possibilities and retransmission times. For another example, a central controlled scheduling algorithm is designed in [22] to determine the scheduling decisions to reduce collision chances.

In addition to the above works, there also exist works proposing distinct video distribution solutions for vehicles. The work [23] enabled RSUs to share file content cached on them with each other to improve file downloading services. However, they failed to fully explore the caching capacity of all RSUs by only selecting a number of representative RSUs for caching. The work [2] studied a different application, *i.e.*, real time video transmission, in vehicular networks. Frame skipping and transcoding are adopted to handle situations with network congestion.

2.2 Video Content Replication

Replicating video content on devices (*e.g.*, RSUs in vehicular networks) close to end users is a popular approach to improve video service efficiency.

The crowdsourcing-based architectures are described by the works [24], [25], [26] which place file content on edge devices to improve file distribution capacity. The work [10] considered a case with a single server and multiple helpers serving file downloading for moving users. Each helper has limited capacity to cache some most popular files. Thus, a scheme is designed to determine files cached on each helper based on file popularity.

Likewise, video files can be replicated on RSUs in vehicular networks. The work [5] proposed to replicate delay-tolerant contents on RSUs by proposing an efficient distributed replication algorithm based on the content popularity, vehicle-AP contact patterns and content availability. In another case, each user can upload some files to RSUs to broadcast them to vehicles with interests [1]. Applying machine learning techniques for video replication to serve mobile users has been explored in the work [27]. The work [28] synthetically utilized historical information to predict user request distribution in vehicular networks, but caching encoded content is not covered. The work in [29] came up with a novel multimedia streaming framework for ad hoc vehicular networks in information centric networks.

2.3 Network Coding

Applying network coding [12] to improve video streaming performance has been extensively studied for many years.

It has been theoretically proved that network coding techniques have the potential to improve multicast rate [11]. In the light of the benefits brought by network coding,

2. Independent chunks can complement each other.

3. To transrate encoded chunks, we need to recover the original video file first, and then transrate the original file to the requested version. This operation will waste too much resources for an RSU merely replicating a small fraction of chunks for each video.

4. It is worthy to mention the difference between transrating and network coding. Transrating operates original video chunks to yield other versions with lower bitrates for playback. Network coding mixes chunks with randomly generated coefficients to generate encoded chunks for replication and transmission, which however should be decoded before playback.

various works have been dedicated to apply network coding for video distribution. The works [14], [30], [31] have investigated how to apply network coding techniques to improve the efficiency for Peer-to-Peer (P2P) video streaming services. Another protocol is designed using network coding to reduce content transmission delay in P2P networks so as to support video conferencing in real time [32]. By improving energy efficiency to execute coding algorithms, the work [33] has implemented network coding based data transmission on mobile phones and pads for multimedia content sharing. The work [34] has designed the coding scheme to support P2P VoD streaming services. The encoding strategy for video replication in P2P VoD systems which can tradeoff encoding/decoding cost and replication cost was designed by the work [35]. The work [36] designed a novel scheme by placing encoded chunks on edge devices to accelerate file downloading. A stochastic model is established to theoretically prove the performance gain achieved by the new scheme. A survey paper has introduced existing approaches for multimedia content distribution with network coding in P2P networks [37]; while a broader survey on the multimedia content distribution with network coding under various network conditions was conducted by the work [38].

In a word, our work is different from aforementioned works because we improve the video distribution performance by reducing the caching space occupied by each replicated video on RSUs so that more videos can be replicated on RSUs. Moreover, our strategy considers this problem from a new perspective and thus can be implemented together with some existing video distribution strategies in vehicular networks to further enhance video transmission efficiency.

3 Background and Preliminary Modeling

This section presents the system model to be studied. We provide examples to illustrate chunk download processes and introduce preliminary knowledge for analysis.

3.1 System Overview

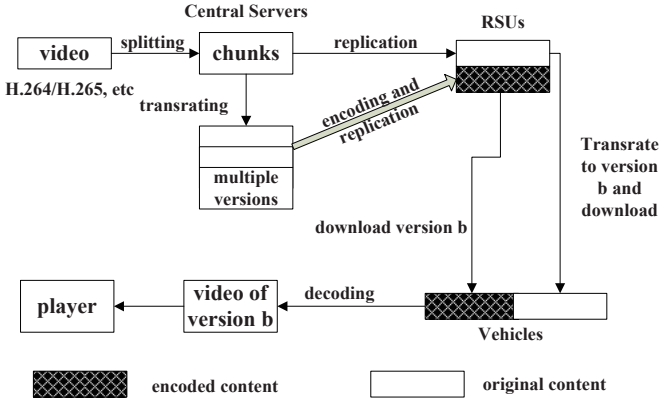


Fig. 1: The overview of the video distribution system in vehicular networks

The overview of the video distribution service in our study is presented in Fig. 1. Videos encoded with H.264/H.265 can be split into multiple *original* chunks with equal size. Original chunks are further transrated into multiple versions with different bitrates to fit in users' playback devices. By applying network coding such as random linear codes to original chunks, an infinite number of encoded chunks can be generated [15], [31]. Splitting a video into chunks is a popular approach for video distribution. According to previous works [14], [30], [31], [39], chunk splitting and assembling operations can be implemented on the Application layer. Then, a fraction of all chunks are replicated on each RSU for vehicles to download. Note that original chunks are transrated to multiple versions with different bitrates before they are encoded for replication. After collecting a sufficient number of independent chunks, the original video file (of the requested version) can be decoded for playback. Here our strategy is independent from the video encoder (*e.g.*, H.264) in use, and we will not specify video encoders any more in the rest content. To make our presentation uncluttered, encoding only refers to encoding original chunks with network coding hereafter. Chunks that are not encoded by network coding are called original chunks.

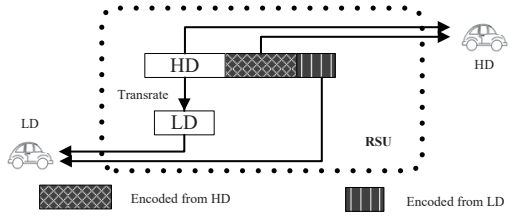


Fig. 2: An example to illustrate the case that an RSU serving two different video versions.

Since our work focuses on the content transmission from RSUs to vehicles, we zoom in to further observe the download process through an example in Fig. 2 with an RSU serving two different video versions: high definition (HD) and low definition (LD). To figure out the replication strategy to achieve the best download performance, we analyze this process with three steps: 1) Analyzing the download performance by fixing the space occupied by original or encoded chunks; 2) Optimizing the space size allocated to original or encoded chunks; 3) Discussing how to support video streaming and the limitations of network coding.

3.2 Benefit of Network Coding

We explain how network coding accelerates chunk downloading processes with two examples.

Two concrete examples are given in Fig. 3 and Fig. 4 respectively. In the first example, only original chunks are replicated on RSUs. Each RSU randomly chooses 2 chunks out of all 4 chunks for replication. In this example, it takes three steps for vehicles to retrieve all chunks. In contrast, in Fig. 4, only two steps are taken if RSUs replicate encoded chunks. The encoded chunks can be generated by a central server to guarantee their mutual independence. As the second example shows, chunks a, b, c and d are independent such that four original chunks can be recovered by solving four linear equations. For the general case, a central server can generate numerous encoded chunks by

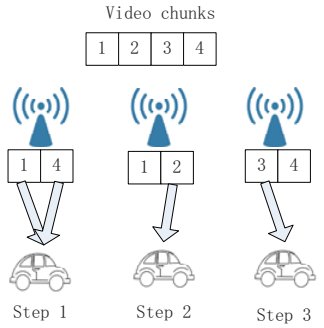


Fig. 3: A case to show the process a vehicle downloads original video chunks from RSUs. There are in total four chunks, while each RSU only replicates two chunks.

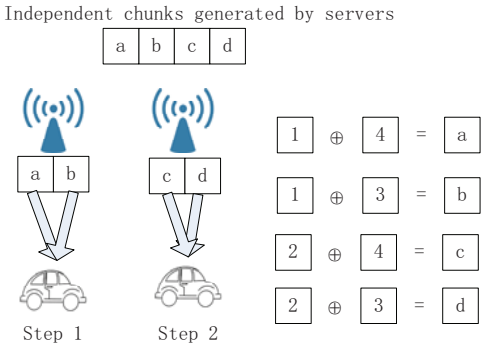


Fig. 4: A case to show the process a vehicle downloads encoded chunks from RSUs. Each RSU replicates two encoded chunks.

XORing (exclusive or) chunks with randomly generated coefficients. For more details, please refer to works [11], [40].

Discussion: Through the above examples, we can see the advantage to replicate encoded chunks on RSUs. However, the cost is the loss of flexibility in chunk transrating in that an encoded chunk could be a combination of multiple original chunks, which cannot be transrated to chunks with a lower bitrate freely. Please check the second example, in which chunk a is the XOR result of chunk 1 and 4, and thus it cannot be directly transrated by RSUs to a new encoded chunk with a lower bitrate. Thus different versions must be maintained for encoded chunks to fit in user devices. Due to the drawback to merely replicate original or encoded chunks, we design the mixed strategy replicating both kinds of chunks on RSUs.

3.3 Video Replication Strategy

Now, we describe chunk replication strategies by setting limited storage space for each RSU.

Let v denote the number of video versions with different bitrates. The original file is with the highest bitrate b_0, which can be transrated to any other version with a lower bitrate if necessary. Unfortunately, an encoded chunk cannot be transrated simply, and an RSU has to maintain all v encoded versions. It implies that $\frac{(b_0+b_1+\cdots+b_{v-1})s}{b_0}$ space will be taken up to completely replicate a chunk, where b_i is the bitrate of version i and s is the size of an original chunk. Here, we assume that all different versions have equal numbers of chunks.

We describe the mixed replication strategy as follows. Let B denote the maximum number of original chunks that each RSU can store, where $B \ll m$. There are totally n RSUs to serve vehicles with $nB \gg m$ so that the aggregate space of all RSUs is sufficient to replicate the whole file. For a video file, an RSU needs to determine w, (*i.e.*, the number of encoded chunks of each version) and l (*i.e.*, the number of original chunks), for replication. Here, the constraint is that $l + w\rho \leq B$, where $\rho = \frac{b_0+\cdots+b_{v-1}}{b_0}$ is the size ratio of v encoded chunks generated for v versions over an original chunk. The l original chunks are randomly selected from total m chunks. The encoded chunks are generated and distributed to RSUs by a central server.

Two special cases of the mixed strategy are achieved by letting either $w = 0$ or $l = 0$. If $w = 0$, it is regarded as the plain random strategy by merely replicating original chunks on RSUs. If $l = 0$, it is regarded as the encoded strategy because all chunks in the system are encoded (with network coding).

3.4 Download Scheduling Scheme

In our problem, a chunk is the unit for downloading. Each vehicle maintains a buffer to cache downloaded original chunks before they are assembled to recover the original file, *i.e.*, decoding. The download scheduling scheme specifies the priority of each useful chunk for downloading when a vehicle contacts an RSU.[5]

The download scheduling scheme for encoded chunks is very trivial. We only blindly download all encoded chunks until m independent chunks are collected so that we can recover the file.

It is worth mentioning that the priority to download original chunks is higher than encoded chunks. Original chunks will be transrated to the requested version before they are transmitted. The transrating process is omitted so that we can focus on the downloading process. If no useful original chunk can be found, vehicles download encoded chunks of the requested version. This simple scheduling scheme maximizes the downloading performance by deferring the download of encoded chunks as much as possible.

3.5 Vehicle Route

For simplicity, a vehicle route can be defined by a series of RSUs from the source to the destination. We assume vehicles will not visit a particular RSU more than once during their trips in that vehicles do not need to travel along the same road more than once to reach their destinations. In simulations, we will relax this assumption by allowing vehicles to revisit RSUs with a very low probability.

Abstracting a vehicle route as a number of RSUs will significantly simplify our analysis. With this abstraction, we

5. Recall that an original chunk is useful if it is independent from all other downloaded chunks.

can focus on the download process of chunks from RSUs and be indifferent to RSUs' locations and vehicles' routes.[6]

3.6 Evaluation Metric

In accordance with our abstraction for vehicles' routes, we use the number of RSUs (also regarded as the number of steps) a vehicle needs to visit to retrieve the entire video file as the metric for theoretical analysis. To avoid the influence of randomness, we calculate the average number of steps for multiple vehicles. The principle is that a faster download process implies that vehicles can finish video downloading by visiting a fewer number of RSUs. We optimize the replication of chunks on each RSU using this metric.

To evaluate our algorithms with more realistic settings, we adopt an additional metric: average file downloading time. This metric is widely used in previous works to evaluate video downloading services.

3.7 Problem Definition

We proceed to define the problem to be studied in our paper. Without loss of generality, we consider the scenario in which a large video file with m chunks is to be distributed to a number of moving vehicles. Vehicles can download at most d chunks from an RSU during their contact periods. Here, we use a fixed d for all RSUs, which will be relaxed later. Decoding will be conducted if m independent chunks are collected by vehicles. Note that we do not discriminate original and encoded chunks for decoding. An original chunk can be perceived as a special encoded chunk.

Consider the problem that a vehicle downloads chunks of the video version with bitrate b from RSUs. Let D_i^b/H_i^b denote the number of original/encoded chunks after visiting i different RSUs. Initial values are $D_0^b = H_0^b = 0$. The file download completes when $D_i^b + H_i^b \geq m$ and our objective is to minimize τ where $D_\tau^b + H_\tau^b \geq m$ but $D_{\tau-1}^b + H_{\tau-1}^b < m$.

Unfortunately, we cannot explicitly express τ as a function of tuneable parameters such as l and w. Therefore, instead of minimizing τ directly, we try to derive the values of D_i^b and H_i^b, and maximize them so as to minimize τ. We notice that they can be solved iteratively. Assuming that D_i^b and H_i^b are known, we have two important insights to derive D_{i+1}^b and H_{i+1}^b.

- $D_{i+1}^b - D_i^b$ is determined by the set of original chunks which have been downloaded by the vehicle, the set of original chunks replicated by the RSU and the download capacity d.
- $H_{i+1}^b - H_i^b$ is determined by the number of encoded chunks replicated by the RSU and the residual download capacity.

Given that RSUs randomly select chunks for replication, D_i^b and H_i^b are random variables. Their distribution is determined by the value of l and w in the replication strategy. Our problem is to maximize $D_i^b + H_i^b$ and hence minimize τ by choosing appropriate l and w with the constraint that $l + \rho w \leq B$.

Since the next section, we will unfold our models to analyze chunk download processes with different values for w and l.

6. Some previous works proposed to organize vehicles close to each other to form an ad hoc network to share partially downloaded content [16], [21], [22] when RSUs are absent. However, connections formed by moving vehicles can be extremely unreliable because of short contact duration. Collisions caused by ad hoc communications may further deteriorate file delivery performance. Thus, the ad hoc scenario is not covered by our work.

4 Homogeneous Case

For simplicity, our analysis begins with a homogeneous scenario, in which each vehicle can at most obtain d chunks from an RSU. This assumption can considerably reduce the state space in our analysis. We proceed with the simplest strategy (*i.e.*, the plain random strategy) merely replicating original chunks on RSUs. Then, we extend our analysis for the mixed strategy that replicates both original and encoded chunks on RSUs.

We study this problem with stochastic models. Considering that a vehicle is downloading chunks of bitrate b, we use the number of chunks downloaded by the vehicle at the time just before it enters the area covered by the next RSU as its state. In other words, the vehicle's state is represented by the number of downloaded chunks (*i.e.*, D_i^b) when the vehicle has visited i different RSUs. Apparently, this is a Markov stochastic process because the number of chunks obtained from RSU i is independent from previous states, but only depends on the current state. To guarantee fairness for vehicles downloading different video versions, their download processes should be identical, which implies that it is unnecessary to discriminate vehicles using their video versions. To keep our notations concise, we just use D_i to represent vehicles' state after they have visited i RSUs. The initial condition is that $D_0 = 0$. Our objective is to derive the probability distribution of D_i.

4.1 Modeling Plain Random Strategy

For the plain random strategy, one has $w = 0$ and $l = B$. Each RSU exactly replicates l original chunks for the video.

The file downloading process depends on not only the chunks replicated on RSUs but also the download capacity. Thus, we need to analyze two separate cases: $d \geq B$ corresponding to the case with unlimited download capacity and $d < B$ corresponding to the case with restricted download capacity.

4.1.1 Case I

For this case, a vehicle can download at most l useful chunks by contacting an RSU.

Let $\vec{\pi}_i = (\pi_{0|i}, \pi_{1|i}, \ldots, \pi_{m|i})$ denote the probability distribution of D_i. The element $\pi_{j|i}$ is the probability to accumulate j chunks at state i. $\vec{\pi}_i$ is determined by $\vec{\pi}_{i-1}$ and the number of chunks downloaded from RSU i. By contacting with a sufficient number of RSUs, D_i will increase to m finally. So, this is a transient stochastic process. To derive $\vec{\pi}_i$, we need to derive the transition matrix denoted by $\mathbf{P}$ first.

Given k_i original chunks downloaded by a vehicle, the probability to find additional k_j useful chunks from a new RSU is

$$Pr(D_{i+1} = k_i + k_j | D_i = k_i) = \frac{\binom{k_i}{l-k_j}\binom{m-k_i}{k_j}}{\binom{m}{l}}, \quad (1)$$

for $0 \leq k_j \leq \min\{l, m-k_i\}$. Obviously, for probability distribution, we have $\sum_{k_j=0}^{\min\{l,m-k_i\}} Pr(D_{i+1} = k_i+k_j|D_i = k_i) = 1$. Then, the element $P_{k_i+k_j,k_i}$ in matrix $\mathbf{P}$ representing the probability jumping from the state with k_i chunks to the state with $k_i + k_j$ chunks with one step is $Pr(D_{i+1} = k_i + k_j|D_i = k_i)$, which is derived from Eq. (1). Otherwise $P_{k_i+k_j,k_i} = 0$ implies that it is impossible to jump from the state with k_i chunks to the state with k_i+k_j chunks at this moment. In this way, all elements in matrix $\mathbf{P}$ can be determined. Together with the initial conditions $\pi_{l|1} = 1$ (because l chunks can be downloaded certainly from step 1) and $\pi_{i|1} = 0$ for $i \neq l$, we can iteratively derive $\vec{\pi}_i$ as follows:

$$\vec{\pi}_i = \mathbf{P}^{i-1} \cdot \vec{\pi}_1. \tag{2}$$

Although we can derive the distribution of D_i for any i with $\mathbf{P}$, initial conditions and Eq. (2), it is too complicated to derive a closed-form solution of $Pr(D_i)$, which impedes our further analysis. To simplify this problem, we turn to derive the expectation of D_i denoted by $E[D_i]$ so that we can conduct a deeper analysis based on its closed form solution.

Let us first consider an elementary probabilistic problem. Given a vehicle with k downloaded chunks and an RSU replicating l original chunks randomly selected from total m chunks, what is the expected number of useful chunks the vehicle can find from the RSU? Note that we can also assume that k chunks owned by the vehicle are also randomly picked out from m chunks because all RSUs also randomly select chunks for replication.

For any particular chunk already obtained by the vehicle, the probability is $\frac{l}{m}$ to find the same chunk from the RSU. Thus, on average, there are $\frac{kl}{m}$ chunks that are dependent and useless for the vehicle. Hence, there are $l - \frac{kl}{m}$ effective chunks that can be downloaded by the vehicle. Based on the above discussion, we can write down the following iterative equation:

$$E[D_{i+1}] \quad = \quad E[D_i] + l - E[D_i]\frac{l}{m}. \tag{3}$$

It is easy to transform the above equation into

$$E[D_{i+1}] \quad = \quad m - (m-l)\left(1 - \frac{l}{m}\right)^i. \tag{4}$$

As i approaches infinity, $E[D_i]$ approaches m. If $m \gg l$, it can be simplified as $E[D_{i+1}] = m - (m-l)e^{-\frac{il}{m}}$. This is a concave function with i if i were a real number. This property indicates that download performance can deteriorate severely with the increase of i.

Eq. (4) also helps us to understand the forte to replicate encoded chunks. By generating chunks via a central server, almost all encoded chunks are independent. The downloading efficiency will not decrease with the download progress. This case is trivial for analysis. By accessing an RSU, the downloaded number of chunks will just increase by l, and the file will be recovered by accessing $\lceil \frac{m}{l} \rceil$ RSUs.

4.1.2 Case II

In this case, we have $d < l$. From Eq. (1), we observe that the range of k_j should be restricted to $0 \leq k_j \leq \min\{d, m-k_i\}$, and all $Pr(D_i + 1 = k_j + k_i|D_i = k_i)$ should be normalized such that their sum equals to 1. The iterative equation (*i.e.*, the expected number of downloaded chunks until step $i+1$) becomes

$$E[D_{i+1}] \quad = \quad E[D_i] + \min\left\{l - E[D_i]\frac{l}{m}, d\right\}. \tag{5}$$

Note that the difference between Eq. (5) and Eq. (3) lies in the constraint that at most d chunks can be downloaded from each RSU. $E[D_i]$ is a monotonic increasing function with i, and the number of initial chunks must be d. Let $\eta = \lfloor \frac{l-d}{d} \rfloor$, it turns out that

$$E[D_{i+1}] = \begin{cases} (i+1)d, & \text{for } i \leq \eta, \\ m + (E[D_\eta] - m)\left(1 - \frac{l}{m}\right)^{i+1-\eta}, & \text{for } i > \eta. \end{cases} \tag{6}$$

By comparing Eq. (3) with Eq. (6), we note that the downloading process of the second case (with $d < l$) can be partitioned into two stages. During the first stage, the number of downloaded chunks increases linearly with the rate d per step; while in the second stage, the increasing rate becomes the same with the rate given in Eq. (4). It is equivalent to prolonging the first stage by lowering d; while the trouble in the second stage persists. The drop of the increase rate for $E[D_i]$ is caused by redundant chunks replicated by the plain random strategy.

Note that $E[D_{i+1}]$ in Eq. (4) and Eq. (6) is always strictly less than m, which can be explained as follows. $E[D_i]$ is the expected number of chunks downloaded after i steps, bounded by m even if i approaches infinity. In practice, it is likely for a vehicle to complete the file downloading with a finite number of steps, which seems contradict with our result. In fact, $E[D_i]$ is the expected value of D_i. In the worst case, all n RSUs cannot recover the file even if $n \gg m$ because there always exists a nonzero probability that n RSUs cannot provide m independent chunks by randomly choosing chunks for replication. A method to avoid the occurrence of this extreme event is to replicate some encoded chunks on RSUs.

4.2 Modeling Mixed Strategy

To get rid of the drawback caused by replicating redundant original chunks, each RSU only replicates l original chunks, while the remaining space will be allocated to store w encoded chunks for each video version. There are total v versions, then the constraint $l + w\rho \leq B$ must be satisfied, where B is the number of original chunks the RSU can store at most for this file.

Analogous to the analysis of plain random strategy, there are two cases which should be analyzed separately (*i.e.*, $d \leq w$ and $d > w$).

Recall that the request scheduling scheme always downloads useful original chunks prior to the download of encoded chunks. Thus, the complication of the mixed strategy lies in the interaction between two subprocesses (*i.e.*, download process of original and encoded chunks). To account for contributions of original chunks and encoded chunks respectively, we let $E[H_i]$ denote the expected number of encoded chunks obtained until step i. By abusing notation a little bit, we let τ denote the expected number of steps vehicles need to complete file downloading. Then, $E[D_\tau]$ and $E[H_\tau]$ are the expected numbers of original and encoded chunks at last.

4.2.1 Case I

This case means that each RSU replicates plentiful encoded chunks to saturate vehicles' download capacity. In this scenario, it is certain that a vehicle can always download d chunks (either original or encoded) from each RSU, and the complete file will be recovered once a vehicle goes through $\lceil \frac{m}{d} \rceil$ different RSUs.

Given $\tau = \lceil \frac{m}{d} \rceil$, it is easy to derive $E[D_\tau]$ and $E[H_\tau]$, which can be further analyzed by two cases.

According to the chunk scheduling scheme described in Sec. 3, if $l \leq d$, a vehicle can always search for useful chunks from l original chunks first before the vehicle downloads encoded chunks with residual download capacity. And in total it can always obtain d chunks. Thus, from Eq. (4), we have

$$\begin{aligned} E[D_\tau] &= m - (m-l)\left(1 - \frac{l}{m}\right)^{\tau-1}, \\ E[H_\tau] &= m - E[D_\tau]. \end{aligned} \tag{7}$$

If $l > d$, a vehicle will try to download d original chunks first. The download subprocess of the original chunks is the same as the case we have analyzed with plain random replication strategy. The $E[D_\tau]$ can be obtained by substituting $i+1 = \tau$ back to Eq. (6), and $E[H_\tau]$ can be calculated with the formula $m = E[D_\tau] + E[H_\tau]$.

4.2.2 Case II

In this case, we have $d > w$ which implies that a vehicle has the capacity to download more than w chunks from an RSU. The number of accessed RSUs is in between $\lceil \frac{m}{d}, \frac{m}{w} \rceil$ to recover the original file. This case is more complicated in that analysis of $E[H_i]$ is more difficult in comparison with the last case. Again, this process can be analyzed with two cases: $\max\{w, l\} < d \leq l + w$ and $w < d \leq l$.

Recall that $E[H_i]$ denotes the expected number of encoded chunks downloaded after i steps. Then, if $\max\{w, l\} < d \leq l + w$, a vehicle will attempt to download l original chunks first before it downloads encoded chunks, and we have

$$E[D_{i+1}] = E[D_i] + l - E[D_i]\frac{l}{m}, \tag{8}$$

$$E[H_{i+1}] = E[H_i] + \min\left\{w, d - l + E[D_i]\frac{l}{m}\right\}. \tag{9}$$

The initial conditions are $E[D_1] = l$ and $E[H_1] = d - l$. With the two iterative equations, we need to solve the smallest i such that $E[D_i] + E[H_i] \geq m$.

If $w < d \leq l$, a vehicle can only download no more than d original chunks. The iterative equations should be modified as:

$$E[D_{i+1}] = E[D_i] + \min\left\{l - E[D_i]\frac{l}{m}, d\right\}, \tag{10}$$

$$E[H_{i+1}] = E[H_i] + \min\left\{w, d - l + E[D_i]\frac{l}{m}\right\}. \tag{11}$$

The initial conditions are $E[D_1] = d$ and $E[H_1] = 0$. Once again, to derive τ, we need to find the smallest i such that $E[D_i] + E[H_i] \geq m$. For $E[H_\tau]$, it is difficult to derive its closed form solution, but the calculation of τ can be obtained with numerical methods. With initial conditions, $E[H_\tau]$ can be computed within $O(m/\min\{d, w\})$ iterations.

4.2.3 Discussion

In summary, in case I, RSUs have replicated plentiful encoded chunks as the backup such that vehicles can complete file downloading by exactly visiting $\lceil \frac{m}{d} \rceil$ RSUs. In case II, we have $d > w$. Although the encoded content is not sufficient to saturate vehicles' download capacity, it provides a lower bound by enabling a vehicle to download at least w encoded chunks from each RSU.

Note that Eq. (8) (or Eq. (10)) is exactly the same as Eq. 3 (or Eq. 5) derived for the case with plain random strategy. This result is straightforward because original chunks are always associated with higher priority for downloading. However, the entire download process is determined by the sum of $E[D_i]$ and $E[H_i]$. $E[D_i]$ and $E[H_i]$ are negatively correlated so that if $E[D_i]$ drops $E[H_i]$ will increase. Thus, the aggregated download can progress steadily. That is the reason why the mixed strategy outperforms the plain random strategy.

5 Algorithm Design

In this section, we optimize the mixed strategy by varying l and w so as to achieve the best performance.

Given storage space B (in terms of original chunks), our objective is to find w^* and l^* to minimize τ (*i.e.*, the number of steps to complete file downloading). However, we confront two challenges to solve this problem:

- τ is a function of l and w, but its expression is too complicated to derive.
- The function τ cannot be differentiated with respect to either l or w. Moreover, τ must be an integer.

To make the problem tractable, we make a compromise by assuming that l and w are real numbers. Instead of deriving the function $\tau(l, w)$, we resort to analyzing $E[D_i] + E[H_i]$. For convenience, we let $y_i = g(i, l, w) = E[D_{i,l}] + E[H_{i,w}]$ denote the expected number of chunks downloaded until step i, where y_i is a function of i, l and w. Here, $E[D_{i,l}]$ (or $E[H_{i,w}]$) is the expected number of downloaded original (or encoded) chunks by replicating l original (or w encoded) chunks on each RSU. Our problem is converted to finding l^* and w^* maximizing y_i.

5.1 Asymptotic Analysis

If the storage space is large enough, *i.e.*, $B \geq v \times d$ where v is the number of video versions and d is the download capacity, there exists a trivial strategy to minimize τ by replicating vw encoded chunks on each RSU. The analysis of this case is straightforward since the storage space is sufficient to saturate vehicles' download capacity by purely replicating encoded chunks. The fact that each RSU just replicates d encoded chunks for each version implies $\tau = \frac{m}{d}$, which is the minimum downloading time we can achieve.

Hereafter, we assume $B < v \times d$ and $w < d$ (*i.e.*, the storage space is limited). We temporarily relax the constraint to download exact m independent chunks by allowing vehicles to download more than m useful chunks. Our objective

is to maximize $y_i = g(i, l, w)$ for all i. Here y_i could be larger than m.By leveraging Lagrange Multiplier, we have:

$$\begin{aligned} \max_{0\le l\le B} \quad & y_i = E[D_{i,l}] + E[H_{i,w}] - \lambda(l + w\rho - B), \\ s.t. \quad & l + w\rho \le B, \\ & \textit{for } i = \eta+1, 2, \ldots, \frac{m}{B/\rho}. \end{aligned} \tag{12}$$

$\frac{B}{\rho}$ is the maximum number of encoded chunks of each version we can replicate on an RSU, based on which we can derive an upper bound for τ. $\eta = \lfloor \frac{l-d}{d} \rfloor$ is the number of steps during which vehicles can certainly download d useful chunks. Here, we assume $l > d$. If $l \le d$, the only modification is to let $\eta = 0$.

Note, at step i, both $E[D_{i,l}]$ and $E[H_{i,w}]$ can be differentiated with respect to l and w. Given the expression of $E[D_i]$ in Eq. (6), we have

Lemma *1.* For $i > \eta$,

$$\frac{\partial E[D_{i,l}]}{\partial l} \approx \frac{1}{m}(i-\eta)(m - E[D_\eta])e^{-\frac{l}{m}(i-\eta)}. \tag{13}$$

Here, $\eta = \lfloor \frac{l-d}{d} \rfloor$, and $E[D_{i,l}]$ is an increasing concave function with l.

The detailed proof is in appendix. In a nutshell, it is proved based on Eq. (6).

Differentiating $E[H_{i,w}]$ with respect to w can be analyzed as follows. By increasing Δ encoded chunks, vehicles can almost download Δ more chunks from each RSU visited after step η. Thus,

$$\frac{\partial E[D_{i,w}]}{\partial w} \approx i - \eta, \tag{14}$$

for $i > \eta$.

Obviously, based on Eq. (13) and Eq. (14), to maximize y_i, KKT conditions should be satisfied. Hence, $l + w\rho = B$ and

$$\frac{\partial E[D_{i,l}]}{\partial l} = \frac{1}{\rho}\frac{\partial E[H_{i,w}]}{\partial w}. \tag{15}$$

Through Eq. (15), we can find l_i^* to maximize y_i, which is

$$l_i^* = \frac{m}{i-\eta} \ln \frac{\rho(m - E[D_\eta])}{m}, \tag{16}$$

$$w_i^* = \frac{B - l_i^*}{\rho}. \tag{17}$$

Here, $\eta = \lfloor \frac{l-d}{d} \rfloor$. If $d \ge l$, we need to set $\eta = 0$. Then l^* and w^* should be rounded to the nearest integer.

Until now, the value of τ is still unknown. Furthermore, τ cannot be differentiated to analyze its monotonicity. This fact implies that even if we have Eq. (16) and Eq. (17), it is still not trivial to decide space partition. Thus, we propose a binary search algorithm to determine l^* and w^*.

5.2 Iterative Space Partition Algorithm

According to our previous analysis, it is certain that $y_i^{max} < y_{i+1}^{max} < \ldots$ because more independent chunks will be accumulated with more steps. Here y_i^{max} is the maximum value of y_i by substituting l_i^* and w_i^* back to $y_i = g(i, l, w)$. By leveraging this property, we can design an algorithm to locate l^* and w^* such that y_τ^{max} is the closest one to m.

The principle of our algorithm is illustrated as follows. From Eq. (16), we can determine l^* and w^* easily. Then, y_i^{max} can be computed correspondingly. Its detailed computation will be described later. If $i > \tau$, y_i^{max} will exceed m, and hence we can assert that $\tau < i$. Otherwise if $y_i^{max} < m$, we have $\tau > i$. The lower bound of τ is $\frac{m}{B}$ (with a single video version) and the upper bound is $\frac{mv}{B}$ (with v versions). Thus, we can halve search space through each iteration.

y_i^{max} can be computed either via the iteration equations we have introduced in the last section or with a very simple way introduced as follows. If $l \le w$, $y_i^{max} \approx wi$ because vehicles can straightly download w chunks from each RSU. Otherwise, we need to first find the largest i' satisfying the inequality $E[D_{i'}] - E[D_{i'-1}] \le w$ (*i.e.*, i' is the step when the increase rate of original chunks is less than w). Then, $y_i^{max} = E[D_{i'}] + (i - i')w$.

The detailed algorithm named as Iterative Space Partition (ISP) Algorithm is presented in Algorithm 1.

Algorithm 1: Iterative Space Partition (ISP) Algorithm

Data: Given storage space B, total number of chunks m, bitrates $b_0, \ldots, b_{v-1}$ of v versions and download capacity d per RSU.
Result: l^* the number of original chunks and w^* the number of encoded chunks for each version
Calculate $\rho = \frac{b_0 + \cdots + b_{v-1}}{b_0}$;
Initialization: Let $low = \frac{m}{B}$, $high = \frac{m\rho}{B}$, $i = \lfloor \frac{hihg+low}{2} \rfloor$;
if $B \ge \rho d$ **then**
  Return $l = 0$ and $w = d$
while $high - low > 2$ **do**
  calculate l^* and w^* from Eq. (16) and Eq. (17);
  calculate y_i^{max};
  if $y_i^{max} > m$ **then**
    $high = i$;
    $i = \frac{i+low}{2}$;
  else
    $low = i$;
    $i = \frac{i+high}{2}$;
Rounding w^* to the nearest integer and $l^* = B - w^*\rho$;
Output: w^* and l^*.

The time complexity of the ISP algorithm is $O(\log_2 m)$, where m is the total number of chunks. The time complexity can be analyzed as below. The calculation of y_i^{max} with the simple method only needs a constant number of steps. The search space ranges from 1 to m. The search starts with the middle point of the search range and halves search space in each round. Thus, the overall time complexity is $O(\log_2 m)$. This algorithm is very efficient. Even if we have 10^4 chunks, the optimal partition point can be located instantly.

6 Heterogeneous Case

The above analysis assumes that RSUs are homogeneous so that we can fix d, (*i.e.*, vehicles' download capacity from each RSU). However, d is a dynamic variable in real systems affected by many factors. Here, we briefly discuss how these factors influence d and propose a simple method to approximate d in the ISP algorithm.

6.1 Multiple Files

To satisfy diversified user interests, vehicular networks have to provide distribution services for multiple video files. However, due to very limited storage space configured with individual RSUs, one needs to determine what files to be replicated on RSUs. Intuitively, file replication decisions should be made according to file popularity distribution [1]. Popular files with more user requests should be replicated on more RSUs, and vice verse.

Once file replication decision is made, B is fixed for each file and we can then apply the ISP algorithm to determine the space allocation between encoded and original chunks. In other words, whether or not to replicate a file is out of the scope of this paper. Our study is designed for these videos RSUs have decided to replicate.

6.2 Road Condition and Network Condition

Another important aspect is vehicles' varying download speeds which can be measured in terms of the number of chunks downloaded from each RSU. Download speed variation implies that d is not a fixed constant any more. For brevity, we summarize three critical factors that can dramatically change d as follows:

1) *Vehicle Velocity*: The sojourn time vehicles spend with an RSU is heavily determined by the vehicle velocity. With longer sojourn time, d should be larger since vehicles can download more file chunks.
2) *Bandwidth Resource*: Available bandwidth resource is a fluctuating variable affected by the number of competing vehicles, channel conditions, the distance between vehicles and RSUs and interference between vehicles. Vehicles obtaining more bandwidth resource will have larger d.
3) *Revisit Frequency*: Although it is rare, it is still possible for vehicles to revisit an RSU within a file download session. d will be smaller, if vehicles revisit a particular RSU multiple times.

6.3 Approximating d

From the above discussion, we can conclude that it is not easy to estimate d accurately, hence we cannot set a fixed d in advance. What is more, the value of d could be very dynamic that can instantaneously change with network states. Not only the model with heterogeneous d's will be extremely complicated, but also the result is unreliable because of its dynamics.

From another perspective, the performance gain achieved by the mixed strategy should not be very sensitive with d. The mixed strategy improves video distribution performance because encoded chunks can reduce replication redundancy. Hence, we infer that video distribution performance can be enhanced with a reasonable approximation of d. Thus, we propose to set d just as the estimated average number of chunks a vehicle can download from RSUs, which is denoted by $\hat{d}$. We assume there exists a central server that can collect some operation information from each RSU, then $\hat{d}$ can be estimated from operation logs with history download speeds of vehicles. For example, each RSU can keep a variable to record the average number of chunks vehicles can download from it. The central server can compute the average value of the entire system $\hat{d}$ by averaging numbers reported by all RSUs. To validate the effectiveness of $\hat{d}$, we conduct trace based simulations in Sec. 8 by using $\hat{d}$ in the ISP algorithm.

7 Discussion

In this section, we examine two more issues when applying our strategy in practical systems. We discuss how to extend our strategy to support video streaming and the limitations of network coding in our strategy.

7.1 Extension to Support Video Streaming

Video streaming is a more complicated video distribution service than video downloading. However, video downloading strategy forms the basis for video streaming services .

There is a large body of previous works that have investigated how to leverage network coding for video streaming. According to prior works, video streaming can be considered as a sequence of video segments downloading in playback order. Inspired by previous works, we extend our strategy to support video streaming services by splitting each video file into multiple video segments, which can be played sequentially. Each video segment is replicated and served independently. Vehicles only need to download segments out of the same video according to the playback order. In other words, a video segment will be downloaded with higher priority if it is closer to its playback deadline. In Fig. 5, we illustrate how segments are downloaded from RSUs. Each segment can be decoded and sent to the player independently. This extension is very similar to algorithms presented in previous works. Due to limited space, we will not give detailed streaming algorithms. For more discussion, please refer to [14], [30], [31].

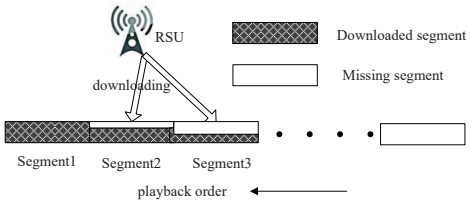


Fig. 5: Providing video streaming services by sequentially downloading video segments .

7.2 Limitation of Network Coding

We move on to discuss the overhead caused by the limitations of network coding.

From users' perspective, the overhead is mainly caused by decoding chunks. Decoding efficiency depends on the computing performance of users' devices and the specific design of the encoding algorithm (*e.g.*, field size). It is easy to understand that devices with higher computing performance can complete decoding faster. It is more involved to explain the influence of the coding design. To generate an

infinite number encoded chunks, random linear codes cannot guarantee all encoded chunks are independent.[7] Given total m original chunks, vehicles need to collect $m(1+\epsilon)$ chunks to recover the original video, where ϵ is a very small positive number depending on field size (which is the space to generate coefficients for encoding). Intuitively speaking, with larger field size, ϵ will be smaller but longer decoding time is needed, and vice versa.

In short, the overhead of network coding will lower video downloading performance a little bit. It is not easy to theoretically quantify this influence because it is difficult to model computing performance of users' devices and the design specifications of the encoding algorithm in our study. Instead, we study the overhead caused by the limitations of network coding via simulations in the next section.

8 Performance Evaluation

In this section, we implement a simulator to verify theoretical models, and illustrate the performance gain achieved by adopting our strategy. The source files of the simulator are publicly available at [41].

8.1 Numerical Simulation

8.1.1 Simulation Setting

Since the purpose is to verify the correctness of models, the simulator is configured with the same settings as the description of our models. To simulate a large system, we set $n = 5000$ RSUs. The other parameters are set by default as follows. The video file is split into $m = 5000$ chunks with 1.0Mb for each chunk. According to previous works, the playback of each chunk should last 1-10 seconds [39]. Typically, a large video can be split into thousands of chunks. Since our analysis is independent from video content we use a dummy file for simulation so that we can adjust bitrates and chunk size flexibly. Since our strategy aims to improve storage efficiency, B should be set as small as possible. Hence we set $B = 100$ implying that each RSU only needs $\frac{B}{m} = 2\%$ space to replicate a video. We set $\rho = 2$ by assuming there are three video versions with bitrates $b_0 = 1.0$Mbps, $b_0 = 0.7$Mbps and $b_2 = 0.3$Mbps. Regarding the mixed strategy, the optimal replication strategy is represented by l^* and w^* which are determined by the ISP algorithm. For replication, each RSU randomly selects l original chunks out of $m = 5000$ chunks and w encoded chunks for each video version.

By default, a new vehicle route will be generated prior to the execution of each simulation. Since all RSUs make replication decisions independently, the next RSU to be accessed is randomly selected from the remaining unvisited RSUs. The chunk scheduling scheme is implemented as we have introduced in Sec. 3, which always downloads (transrated) original chunks with higher priority. Each simulation case is repeated for 100 times to get the average performance.

7. For large-scale systems, the population of encoded chunk is very large and can be considered as an infinite number.

8.1.2 Verifying Theoretical Model

We first present three figures to show the accuracy of our models, and then we exhibit results to illustrate how each parameter affects downloading performance. At last, we compare the mixed strategy with the other two strategies.

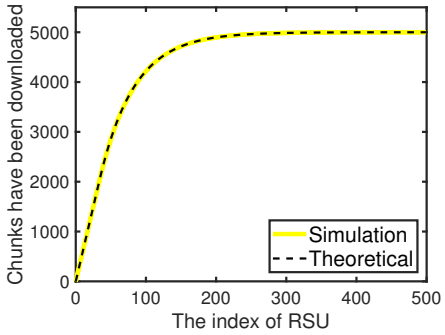


Fig. 6: The simulation result to verify the stochastic model with plain random strategy by setting $m = 5000$, $w = 0$, $l = 100$, $d = 60$, $v = 3$ and $\rho = 2$.

Figure 6 presents the simulation result by setting $m = 5000$, $w = 0$, $l = 100$, $d = 60$, $v = 3$ and $\rho = 2$. This simulation corresponds to the case studied in Eq.(6). In Fig. 6, x-axis represents the sequence ID of visited RSUs (*i.e.*, download steps) and y-axis represents the average number of chunks downloaded by 100 vehicles. we plot both the simulation and theoretical curves in the figure. The accuracy of our model is indicated by the negligible gap between two curves. Note that both curves increase rapidly when RSU index is less than 100. However, this strategy encounters difficulty to download useful chunks when RSU index exceeds 100. This result is consistent with our analysis in Sec. 4, which is caused by the redundancy of original chunks.

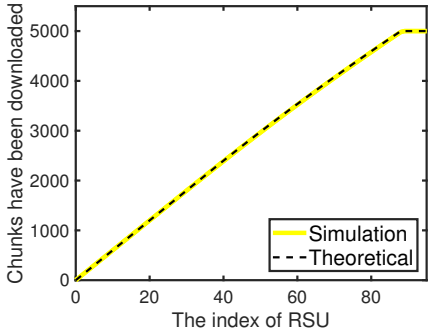


Fig. 7: The simulation result to verify the stochastic model with the mixed strategy by setting $m = 5000$, $w = 30$, $l = 40$, $d = 60$, $v = 3$ and $\rho = 2$.

We simulate the mixed strategy by assuming there are three versions (*i.e.*, $v = 3$) with $\rho = 2$ to verify Eq. (8) and Eq. (9). Here, we set $m = 5000$, $w = 30$, $l = 40$ and $d = 60$. Again, the x-axis is the index of visited RSUs and the y-axis is the average number of downloaded chunks (both original and encoded). If the total number of chunks exceeds 5000, the download process terminates. We can observe that

the simulation curve and theoretical curve almost overlap indicating the accuracy of our models. It is interesting to note that both curves increase almost linearly. There is no performance degrade phenomenon as in Fig. 6. This result illustrates the effectiveness of the mixed strategy though l and w are not optimized.

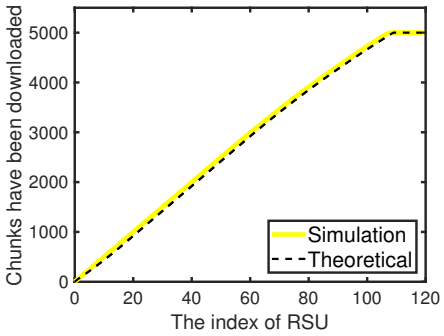


Fig. 8: The simulation result to verify the stochastic model with the mixed strategy by setting $m = 5000$, $w = 20$, $l = 60$, $d = 50$, $v = 3$ and $\rho = 2$.

In Fig. 8, we repeat the simulation of the mixed strategy by modifying parameters as $m = 5000$, $w = 20$, $l = 60$ and $d = 50$. This simulation corresponds to Eq. (10) and Eq. (11). In this case, the download capacity is restricted as the bottleneck and it is expected more steps are required to complete video file downloading. The meanings of x-axis and y-axis are the same as those in the last simulation. We can observe that the theoretical result is precise because two curves are very close. The two curves increase steadily to about $m = 5000$ when RSU index approaches 110 manifesting the effectiveness of the mixed strategy in this scenario.

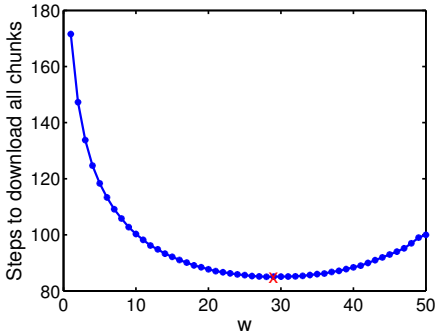


Fig. 9: Simulation results comparing performance of different replication strategies by varying w from 1 to 50 and fixing $m = 5000$, $B = 100$, $d = 100$, $v = 3$ and $\rho = 2$.

In Fig. 9, we present the simulation results of different replication strategies by varying w from 1 to 50 with $m = 5000$, $B = 100$, $d = 100$, $v = 3$ and $\rho = 2$. Recall that w is the number of encoded chunks replicated for each version. Varying w from 1 to 50 is equivalent to simulating 50 different mixed strategies. The y-axis is the average number of steps required to complete file downloading. It is worth noting that the best performance is achieved when $w = 29$. Either too small or too large w cannot attain very good performance. This result reveals the importance to optimize l and w. In fact, enumerating w as this simulation shows is a brute force method which is inefficient and not scalable with the system scale. The w^* determined by the ISP algorithm is marked with a cross in the figure which almost overlaps with the optimal value.

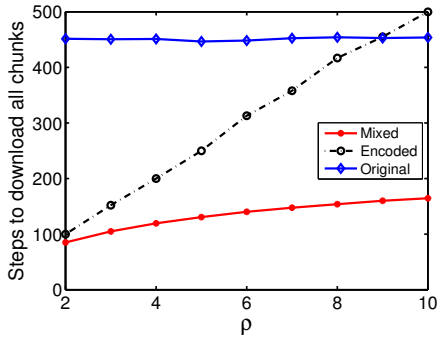


Fig. 10: Simulation results comparing performance of different replication strategies by varying ρ from 2 to 10 and fixing $v = 2\rho - 1$, $m = 5000$, $d = 100$ and $B = 100$. w and l are determined by the ISP algorithm.

ρ	2	3	4	5	6	7	8	9	10
l	42	55	60	65	70	72	76	73	70
w	29	15	10	7	5	4	3	3	3

TABLE 1: The value of l and w of the mixed strategy in Fig. 10.

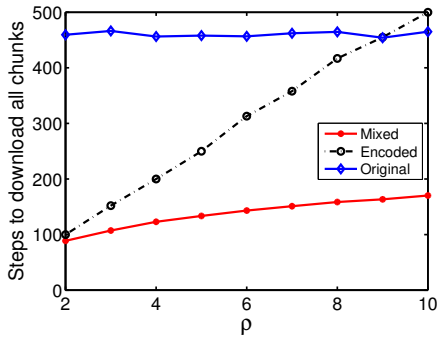


Fig. 11: Simulation results comparing performance of different replication strategies by varying ρ from 2 to 10 and fixing $v = 2\rho - 1$, $m = 5000$, $d = 60$ and $B = 100$. w and l are determined by the ISP algorithm.

ρ	2	3	4	5	6	7	8	9	10
l	40	55	60	65	64	65	68	73	70
w	30	15	10	7	6	5	4	3	3

TABLE 2: The value of l and w of the mixed strategy in Fig. 11.

According to Eq. (16) and Eq. (17), ρ is the key parameter determining the optimal l and w. To visualize the influence of ρ, we compare the average number of steps to complete file downloading with different replication strategies by varying ρ from 2 to 10 in Fig. 10. According to the definition

of ρ, we have $\rho = \frac{b_0+\cdots+b_{v-1}}{b_0}$. Thus, we cannot set v and the bitrate of each video version arbitrarily once ρ is fixed. In other words, v, $b_0, \ldots, b_{v-1}$ must satisfy the constraint that $\rho = \frac{b_0+\cdots+b_{v-1}}{b_0}$. Meanwhile, it is required that $v > \rho$ because b_0 is the bitrate of the original video which is of the largest bitrate. In our simulations, we set the number of video versions as $v = 2\rho-1$ which can guarantee that $v > \rho$ if $\rho \geq 2$. In fact, one can set v as any other integer values as long as $v > \rho$. Given ρ and v, the bitrate of each version is generated randomly by restricting that $\sum_{i=0}^{v-1} b_i = \rho b_0$.[8] The other parameters are set as $m = 5000$, $d = 100$, $B = 100$. The performance of the plain random strategy is rather stable and poor regardless of ρ which provides service by transrating original chunks to the requested version on-the-fly. The performance of the mixed strategy is the best one and much better than the plain random strategy, though its performance gradually degrades with the increase of ρ in that more versions have to be maintained for encoded chunks. It is interesting to find that the performance of the encoded strategy deteriorates dramatically with the growth of ρ. Its performance is even worse than that of the plain random strategy. This result reveals the weak side of the encoded strategy. The encoded chunks of different versions cannot complement each other resulting in performance deterioration with the increase of ρ (or v).

In Table 1, the value of l^* and w^* of the mixed strategy is presented with ρ. We can observe that the mixed strategy is prone to allocating more resources to replicate original chunks with the increase of ρ. The reason is that it is more efficient to replicate original chunks which can be flexibly transrated to the requested version.

In Fig. 11 and Table 2, we repeat the simulation in Fig. 10 with the same settings except that $d = 60$, which is less than $B = 100$. In Fig. 11, we can observe that the result is very similar to that of Fig. 10 and the mixed strategy still achieves the best performance. Likewise, in Table 2, w^* in the mixed strategy decreases with the increase of ρ as we have indicated in Table 1. Therefore, we can conclude that the performance of the mixed strategy is not sensitive to the value of d.

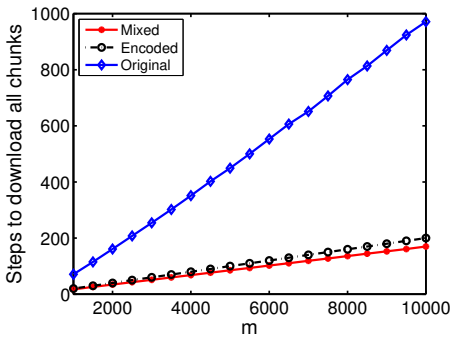


Fig. 12: Simulation results of different replication strategies by varying m from 1000 to 10000 and fixing $d = 100$, $B = 100$, $v = 3$ and $\rho = 2$. w and l are determined by the ISP algorithm.

8. We use the same way to set bitrates as we vary ρ in other simulations.

In this simulation, we change the file size by varying m from 2000 to 10000 with $d = 100$, $B = 100$, $v = 3$ and $\rho = 2$. The results are exhibited in Fig. 12. Again, the mixed strategy is the best one. The performance of the encoded strategy is also satisfactory because we have only three versions, while the plain random strategy is the worst one especially when m is large. The gap between the plain random strategy and the other two strategies expands dramatically with the increase of m. The reason has been shown in Fig. 6. It is more difficult to find useful chunks if vehicles have obtained more chunks. Apparently, vehicles suffer this difficulty more severely with a larger file size.

8.1.3 Embedded with Existing Algorithm

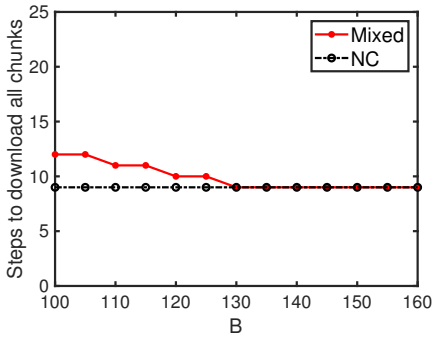


Fig. 13: Implementing the mixed strategy on top of the NC algorithm for video distribution.

As we have stated, the mixed strategy is a way to effectively reduce the storage occupied by each replicated video so as to improve video distribution efficiency. It turns out that the mixed strategy can be embedded into other strategies that mainly focus on transmission efficiency. To visualize this point, we implement the mixed strategy together with the NC strategy proposed by [16], which is applicable for video content distribution for vehicles on highways. Although network coding is adopted by the NC strategy, it assumes that complete files are replicated on RSUs. Due to limited space, we will not introduce the detailed NC strategy. We intend to emphasize that the NC strategy can be improved by replicating partial file content with our mixed strategy.

According to the simulation settings in [16], we simulate 6 vehicles on a highway that are downloading a file with the size 5000Mb which is divided into 5000 chunks. The download amount is at most 100 chunks from a single RSU. Vehicles can also exchange chunks with each other perfectly when RSUs are absent. The metric is the number of steps to complete file downloading. We repeat our simulation 100 times to get the average number. For the NC strategy, the whole file is replicated, while for the mixed strategy we vary the cache size from 100Mb to 160Mb. There are three video versions with bitrates 1.0Mbps, 0.7Mbps and 0.3Mbps. The result as presented in Fig. 13 indicates that the mixed strategy achieves the same performance as NC as long as $B > 130$Mb. It means that the mixed strategy can reduce the consumed cache space from 5000Mb to 130Mb. Considering that the video population is a huge number, the

saved space can be used to replicate other files which will consequentially improve the efficiency of the entire system.

8.2 Trace Based Simulation

8.2.1 Simulation Setting

We proceed to take road conditions and network conditions into account by conducting trace based simulations.

Recall that we abstract vehicles' routes as a number of RSUs. There is no video transmission if RSUs are absent. Thus, road map can be defined through a number RSUs and how vehicles move from an RSU to the next RSU. We generate an undirected graph with 5000 RSUs to represent the map. The map topology is generated according to the reference [42]. According to roads' out-degree in reality, we randomly generate 1 to 4 as the out-degree of an RSU to connect with other RSUs. A transition probability is associated with each edge to represent the probability vehicles transit between two adjacent RSUs. The sojourn time of each vehicle within an RSU obeys exponential distribution with expected value 20 seconds. Communication channels between RSUs and vehicles follow Rayleigh distribution. The download speed switches among ten states, *i.e.*, 500, 600, . . . , 1500 kbps, with the mean value of 1000 kbps [7].

We assume that the request frequency of v video versions is identical. $\hat{d}$ of the system can be estimated with the knowledge of the expected sojourn time, the expected download speed and the expected bitrate of the downloaded file. There are 200 vehicles which randomly select RSUs as their source nodes. A trace will be generated as follows. From the source node, a vehicle randomly selects the next RSU according to the map topology and the transition probability between RSUs. Each generated trace includes 200 RSUs. Note that we do not prohibit vehicles from revisiting the same RSU in trace generation.

8.2.2 Simulating Downloading

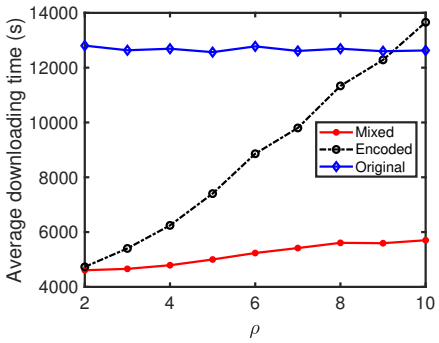


Fig. 14: Trace based simulation comparing the average file downloading time for different replication strategies by varying ρ from 2 to 10, fixing $v = 2\rho - 1$, $m = 5000$ $B = 100$ and dynamic d. w and l are determined by the ISP algorithm.

For video downloading, vehicles can only get chunks from RSUs. We compare the average downloading time to evaluate replication strategies by varying ρ from 2 to 10. Correspondingly, the number of video versions v is varied from 3 to 19 with the relation $v = 2\rho - 1$. For each given v, we generate bitrate distribution randomly by restricting $\sum_{i=0}^{v-1} b_i = \rho b_0$.

ρ	2	3	4	5	6	7	8	9	10
l	36	43	52	55	58	58	60	64	60
w	32	19	12	9	7	6	5	4	4

TABLE 3: The value of l and w of the mixed strategy in Fig. 14.

The average file downloading time of all vehicles is plotted in Fig. 14. Table 3 records the value of l^* and w^* for each ρ.

Here, we would like to emphasize the setting of $\hat{d}$ in our simulation. It is tough to determine w^* if d is dynamic as in the realistic settings because d is required for the ISP algorithm. Fortunately, in Fig. 9, the file downloading performance around w^* is quite flat, thereby we use $\hat{d}$ (*i.e.*, the average number of downloaded chunks from each RSU) to approximate d for the calculation of w^* in Eq. (17).

We can also see from Fig. 14, the mixed strategy achieves the minimum average file downloading time irrespective the value of ρ. The figure pattern is very similar to patterns in Fig. 10 and Fig. 11, which can be explained with the same reason. This essential result indicates the effectiveness using $\hat{d}$, the robustness of the mixed strategy and its applicability in practical systems.

8.2.3 Limitation of Network Coding

It is known that a small fraction of additional encoded chunks should be downloaded to ensure the video file can be recovered successfully, and additional computing time will be taken to decode encoded chunks. In this work, we empirically show that the negative impact caused by the limitations of network coding is very small for the mixed strategy.

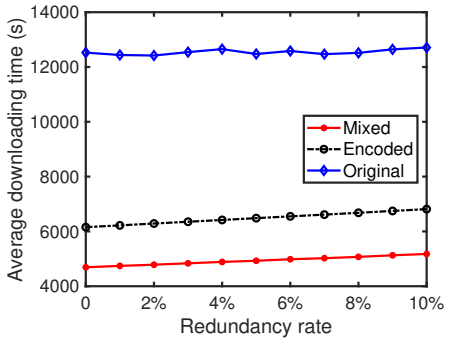


Fig. 15: Trace based video downloading by varying ϵ from 0% to 10% by fixing $\rho = 4$.

According to the theory of random linear codes, $m(1+\epsilon)$ chunks should be downloaded for decoding where ϵ is a very small number depending on the size of field size. Field size is determined by the design of the encoding algorithm. If field size is larger, ϵ will be smaller but decoding time will be longer. To visualize the impact of ϵ, we conduct a trace based simulation by setting $\rho = 4$ with bitrates $1, 0.8, 0.7, 0.6, 0.4, 0.3, 0.2$Mbps. The other settings are the same as those in the last simulation in Fig. 14. As we can see, with the increase of ϵ, vehicles need longer time to complete

file downloading, but the average time is still much smaller than the strategy without encoded chunks.

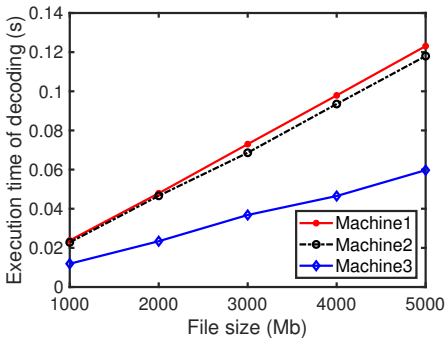


Fig. 16: Decoding time of a file with 5000 chunks on three different computers.

Decoding influence depends on the performance of users' devices. We investigate the influence of decoding overhead by conducting experiment with three computers, which are equipped with Intel(R) Xeon(R) Silver 4114 CPU@2.20GHz, Intel(R) Xeon(R) CPU E5-2640 v4@2.40 GHz, Intel(R) Core(TM) i5-4570 CPU@ 3.20GHz respectively. A public encoding and decoding library is adopted for our experiment.[9] The field size is set as 2^{16}. The experiment result is presented in Fig. 16 by varying file size from 1000Mb to 5000Mb. As we can see, the decoding time increases linearly with file size for all three computers, but the decoding time is always less than 1s. It is evident that the decoding influence is negligible in comparison with downloading time.

8.2.4 Video Streaming

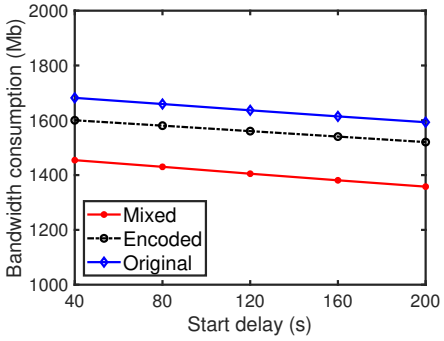


Fig. 17: Extending strategies to support video streaming by splitting the file into 10 segments and varying startup delay from 40s to 200s.

We extend our strategy to video streaming services by splitting the video file with 5000Mb into 10 segments and each segment is further split into 500 chunks. For replication, each segment is regarded as an independent file. To stream the video, segments closer to playback deadline are associated with higher priority. Note that RSUs can only provide services with best efforts for video streaming. In other words, vehicles download chunks from RSUs first. However, remote servers have to be deployed to provide the backup service. If a vehicle cannot obtain a complete segment before its playback deadline, the remote server will deliver the rest content in time. In this setting, it is more appropriate to adopt server bandwidth consumption as the metric to evaluate video streaming services, which is the total amount of video content delivered by remote servers [30].

We conduct a trace-based simulation by setting $\rho = 3$. There are 5 versions with bitrates 1.0Mbps, 0.8Mbps and 0.6Mbps, 0.4Mbps and 0.2Mbps. $B = 100$Mb and each RSU equally divides its storage to replicate segments. It means that we allocate 10Mb space for each segment. In this simulation, we just heuristically set $l = 4$ and $w = 2$ to illustrate that the mixed strategy can be extended to support video streaming.[10] The other settings are the same as the trace based simulation in Fig. 14.

The simulation result is presented in Fig. 17, in which the startup delay is varied from 40 to 200 seconds. Consistent with previous works [14], [30], [31], the average server bandwidth consumption decreases with the increase of startup delay. More importantly, the mixed strategy is the best one consuming the least amount of server bandwidth. This simulation suffices to show the merit of our strategy for video streaming services in vehicular networks.

9 Conclusion

The distribution of large files to vehicles is crucial for the development of vehicular networks. It is an essential functionality supporting a series of important applications, such as video surveillance, video streaming and entertainment, online game, etc. A low quality file delivery service could considerably prohibit the success of vehicular networks. Our work particularly focuses on the case to deliver video files with multiple bitrate versions to vehicles by replicating (encoded) chunks on RSUs. This design significantly outperforms the plain random strategy and the encoded strategy. It can automatically adjust the space allocated to each kind of chunks according to the system settings. This work develops a new design paradigm for vehicle file download services. Other than video downloading, our strategy can also be applied to non-video file downloading, which is a special case of video downloading with only one version. In the future work, we plan to further refine the mixed replication strategy based on the popularity of each video version and load states of each RSU, and refine the video streaming services in vehicular networks.

References

[1] T. H. Luan, L. X. Cai, J. Chen, X. S. Shen, and F. Bai, "Engineering a distributed infrastructure for large-scale cost-effective content dissemination over urban vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, pp. 1419–1435, 2014.

[2] M. A. Bonuccelli, G. G. T+, F. Lonetti, F. Martelli *et al.*, "Real-time video transmission in vehicular networks," in *2007 Mobile Networking for Vehicular Environments*. IEEE, 2007, pp. 115–120.

9. The encoding and decoding library can be accessed from https://github.com/steinwurf/kodo-python, and the library is introduced in [43] .

10. The optimization of l and w for video streaming is a complicated problem that should be analyzed as a different work.

[3] S. Yoon, D. T. Ha, H. Q. Ngo, and C. Qiao, "Mopads: A mobility profile aided file downloading service in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 5235–5246, 2009.

[4] O. Trullols-Cruces, M. Fiore, and J. Barcelo-Ordinas, "Cooperative download in vehicular environments," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 663–678, 2012.

[5] D. Zhang and C. K. Yeo, "Efficient replication for vehicular content distribution," *Vehicular Communications*, vol. 13, pp. 13–26, 2018.

[6] A. M. Mezher and M. A. Igartua, "Multimedia multimetric map-aware routing protocol to send video-reporting messages over VANETs in smart cities," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 611–10 625, 2017.

[7] Z. Deng, Y. Zhou, D. Wu, G. Ye, M. Chen, and L. Xiao, "Utility maximization of cloud-based in-car video recording over vehicular access networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5213–5226, 2018.

[8] R. G. Hollands, "Will the real smart city please stand up? intelligent, progressive or entrepreneurial?" *City*, vol. 12, no. 3, pp. 303–320, 2008.

[9] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through Internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 112–121, 2014.

[10] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[11] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 782–795, 2003.

[12] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[13] C. Gkantsidis and P. R. 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.

[14] M. Wang and B. Li, "R2: Random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, 2007.

[15] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[16] W. Zhu, D. Li, and W. Saad, "Multiple vehicles collaborative data download protocol via network coding," *IEEE Transactions on vehicular technology*, vol. 64, no. 4, pp. 1607–1619, 2015.

[17] K. Liu, J. K. Ng, V. C. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: Vanet as a software defined network," *IEEE/ACM transactions on networking*, vol. 24, no. 3, pp. 1759–1773, 2016.

[18] M. Xing, J. He, and L. Cai, "Utility maximization for multimedia data dissemination in large-scale VANETs," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 1188–1198, 2017.

[19] B. Bellalta, E. Belyaev, M. Jonsson, and A. Vinel, "Performance evaluation of IEEE 802.11 p-enabled vehicular video surveillance system," *IEEE Communications Letters*, vol. 18, no. 4, pp. 708–711, 2014.

[20] B. Hassanabadi and S. Valaee, "Reliable periodic safety message broadcasting in VANETs using network coding," *IEEE transactions on wireless communications*, vol. 13, no. 3, pp. 1284–1297, 2014.

[21] C. Wu, X. Chen, Y. Ji, S. Ohzahata, and T. Kato, "Efficient broadcasting in VANETs using dynamic backbone and network coding," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6057–6071, 2015.

[22] X. Shen, X. Cheng, L. Yang, R. Zhang, and B. Jiao, "Data dissemination in VANETs: A scheduling approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2213–2223, 2014.

[23] D. Zhang and C. K. Yeo, "Enabling efficient wifi-based vehicular content distribution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 3, pp. 479–492, 2013.

[24] L. Chen, Y. Zhou, M. Jing, and T. M. Richard, "Thunder crystal: A novel crowdsourcing-based content distribution platform," in *NOSSDAV, 2015 Proceedings ACM*.

[25] Y. Zhou, L. Chen, M. Jing, Z. Ming, and D. M. Chiu, "Analyzing streaming performance in crowdsourcing-based video service systems," in *LANMAN, 2015 Proceedings of IEEE*. IEEE, 2015.

[26] Y. Zhou, L. Chen, M. Jing, S. Zou, and R. T. Ma, "Design, implementation, and measurement of a crowdsourcing-based content distribution platform," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 5s, p. 80, 2016.

[27] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1046–1061, 2017.

[28] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3520–3535, 2017.

[29] C. Xu, W. Quan, H. Zhang, and L. A. Grieco, "Grims: Green information-centric multimedia streaming framework in vehicular ad hoc networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 2, pp. 483–498, 2018.

[30] Z. Liu, C. Wu, B. Li, and S. Zhao, "Uusee: large-scale operational on-demand streaming with random network coding," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[31] M. Grangetto, R. Gaeta, and M. Sereno, "Rateless codes network coding for simple and efficient p2p video streaming," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, 2009, pp. 1500–1503.

[32] A. Fiandrotti, A. M. Sheikh, and E. Magli, "Towards a p2p videoconferencing system based on low-delay network coding," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012, pp. 1529–1533.

[33] H. Shojania and B. Li, "Random network coding on the iphone: fact or fiction?" in *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*. ACM, 2009, pp. 37–42.

[34] F. Liu, S. Shen, B. Li, B. Li, H. Yin, and S. Li, "Novasky: Cinematic-quality vod in a p2p storage cloud," in *Proc. of IEEE Infocom*, 2011.

[35] Y. Zhou, Y. Xu, and S. Zhang, "Exploring coding benefits in cdn-based vod systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 11, pp. 1969–1981, 2014.

[36] Y. Zhou, T. H. Chan, S. W. Ho, G. Ye, and D. Wu, "Replicating coded content in crowdsourcing-based cdn systems," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[37] B. Li and D. Niu, "Random network coding in peer-to-peer networks: from theory to practice," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 513–523, 2011.

[38] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, "Network coding meets multimedia: A review," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1195–1212, 2013.

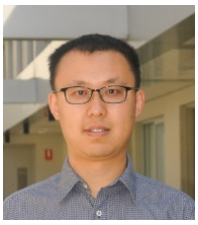
[39] Z. Huang, C. Mei, L. E. Li, and T. Woo, "Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 201–205.

[40] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE transactions on information theory*, vol. 49, no. 2, pp. 371–381, 2003.

[41] "Website link of source files for the simulator," https://github.com/jwc1996/CollaborativelyReplicating, accessed: 2019-09-20.

[42] N. R. Peterson and B. Pittel, "Distance between two random k-out digraphs, with and without preferential attachment," *Advances in Applied Probability*, vol. 47, no. 3, pp. 858–879, 2015.

[43] M. V. Pedersen, J. Heide, and F. H. Fitzek, "Kodo: An open and research oriented network coding library," in *International Conference on Research in Networking*. Springer, 2011, pp. 145–152.



Yipeng Zhou is a lecturer in computer science with Department of Computing at Macquarie University, and the recipient of ARC DECRA in 2018. From Aug. 2016 to Feb. 2018, he was a research fellow with Institute for Telecommunications Research (ITR) of University of South Australia. From 2013.9-2016.9, He was a lecturer with College of Computer Science and Software Engineering, Shenzhen University. He was a Postdoctoral Fellow with Institute of Network Coding (INC) of The Chinese University of Hong Kong (CUHK) from Aug. 2012 to Aug. 2013. He won his PhD degree and Mphil degree from Informatio Engineering (IE) Department of The Chinese University of Hong Kong. He got Bachelor degree in Computer Science from University of Science and Technology of China in 2006.

Jiawen Chen received the B.S. degree from South China Normal University, Guangzhou, China, in 2018. He is currently pursuing the M.S. degree at the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China, under the supervision of Prof. D. Wu. His current research interests include machine learning and multimedia networking.

Guoqiao Ye received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016. He is currently pursuing the master degree in the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, under the supervision of Prof. D. Wu. His research interests include smart grid and computer communications.

Di Wu (M'06-SM'17) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2000, the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2007. He was a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA, from 2007 to 2009, advised by Prof. K. W. Ross. Dr. Wu is currently a Professor and the Assistant Dean of the School of Data and Computer Science with Sun Yat-sen University, Guangzhou, China. His research interests include cloud computing, multimedia communication, Internet measurement, and network security. He was a co-recipient of the IEEE INFOCOM 2009 Best Paper Award. He has served as an Editor of the Journal of Telecommunication Systems (Springer), the Journal of Communications and Networks, Peer-to-Peer Networking and Applications (Springer), Security and Communication Networks (Wiley), and the KSII Transactions on Internet and Information Systems, and a Guest Editor of the IEEE Transactions on Circuits and Systems for Video Technology. He has also served as the MSIG Chair of the Multimedia Communications Technical Committee in the IEEE Communications Society from 2014 to 2016. He served as the TPC Co-Chair of the IEEE Global Communications Conference - Cloud Computing Systems, and Networks, and Applications in 2014, the Chair of the CCF Young Computer Scientists and Engineers Forum - Guangzhou from 2014 to 2015, and a member of the Council of China Computer Federation.

Jessie Hui Wang received the B.S. degree and the M.S. degree in computer science from Tsinghua University, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2007. She is currently an Associate Professor with Tsinghua University. Her research interests include traffic engineering, cloud computing, network measurement, and Internet economics.

Min Chen (SM'09) has been a full professor in the School of Computer Science and Technology at the Huazhong University of Science and Technology since 2012. He is the Chair of the IEEE Computer Society STC on Big Data. His Google Scholars Citations reached 17,000+ with an h-index of 64. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017, and the IEEE Jack Neubauer Memorial Award in 2019. His research focuses on IoT sensing, 5G networks, healthcare big data, and cognitive computing.

APPENDIX

Proof of Lemma 1:

From Eq. (6), if $i > \eta$, we have

$$E[D_i] \quad = \quad m + (E[D_\eta] - m)\left(1 - \frac{l}{m}\right)^{i-\eta}.$$

Here η and $E[D_\eta]$ can be taken as a constant.

If $m \gg l$, then $\left(1 - \frac{1}{m/l}\right)^{m/l} = \frac{1}{e}$ and $E[D_i]$ can be simplified as:

$$E[D_i] \quad \approx \quad m + (E[D_\eta] - m)e^{-\frac{l}{m}(i-\eta)}.$$

Obviously, $\frac{dE[D_i]}{dl} > 0$ and $\frac{d^2E[D_i]}{dl^2} < 0$ for all $i > \eta$. Thus, $E[D_i]$ is an increasing concave function with l.

$$\frac{\partial E[D_{i,l}]}{\partial l} \quad = \quad \frac{1}{m}(i-\eta)(m - E[D_\eta])e^{-\frac{l}{m}(i-\eta)},$$

Here, η in most cases is a small number and $i > \eta$. $E[D_\eta]$ is the number of chunks downloaded when i is no more than $\lfloor \frac{l-d}{d} \rfloor$, which is less than l. ∎

Notation	Meaning
B	buffer size of an RSU to replicate a particular video
l	# of replicated original chunks
l*	# of replicated original chunks achieving optimal performance
w	# of replicated encoded chunks for each version
w*	# of replicated encoded chunks for each version achieving optimal performance
v	# of video versions
b	video bitrate
m	# of original chunks of a video
n	# of RSUs in the system
d	# of chunks vehicles can download at most from an RSU
D_i	# of original chunks downloaded with i RSUs
H_i	# of encoded chunks downloaded with i RSUs

TABLE 4: The list of notations used in the paper.