Replicating Coded Content in Crowdsourcing-based CDN Systems

Yipeng Zhou, Terence H. Chan, Siu Wai Ho, Guoqiao Ye and Di Wu

***Abstract*—Recently, crowdsourcing-based content delivery networks (CDN) emerge as a promising technology that can distribute massive video content to a vast number of Internet users by crawling bandwidth and storage resources from Internet end devices. Any ordinary Internet users with excessive resources can be recruited into such systems as mini-servers. Different from edge servers equipped with dedicated resources in traditional CDNs, the resource of a single mini-server is scarce and volatile that can vary severely with time since its bandwidth is shared by many different applications. How to build a robust high performance crowdsourcing-based CDN system has attracted contributions from both academia and industry, but how to solve the drawback caused by unstable uploading bandwidth is still a challenging problem. So far, a prevalent methodology is to migrate the strategies implemented by traditional CDNs into crowdsourcing-based CDN systems based on the fact that these two kind of systems share many similarities. In this work, our argument is that the content delivery time can be reduced by replicating coded content on mini-servers (which is almost useless for edge servers in traditional CDNs) to enable downloading users to automatically adapt their downloading progress with oscillating bandwidth capacity from different mini-servers. Theoretical model is created to derive the performance improvement (evaluated in term of average file downloading time) achieved by our strategy, which is further validated via simulation. Our work not only provides system designers a more efficient content replication solution, but also can push forward the development of the crowdsourcing-based CDNs.**

***Index Terms*—Crowdsourcing-based CDN, Mini-Servers, Coded Content, Replication .**

I. Introduction

To satisfy the fast growing demands for Internet content, especially for high definition (HD) video content [1], content delivery networks (CDN) have been developed to cache Internet content on edge servers which are close to end users to reduce transmission latency [2], [3]. However, it is pricey to deploy and maintain a large number of dedicated edge servers due to the expensive hardware cost and bandwidth cost. To reduce the cost of CDNs, crowdsourcing-based CDN systems have been designed and implemented by industry that can recruit ordinary end users to join in content delivery game by paying them rewards, e.g., cash [4], [5].

However, the main challenge to run a crowdsourcing-based CDN is from the unstable bandwidth resource crawled from ordinary Internet users. In this work, we introduce a scheme to replicate coded content (which is almost useless in traditional CDN systems) in crowdsourcing-based CDN systems to alleviate the influence caused by unstable uploading resources so as to achieve better content delivery performance. We focus on the downloading service in this paper, and the streaming service will be studied in our future work.

Specifically, crowdsourcing-based CDNs are constructed by motivating ordinary Internet users to contribute their storage resource for content caching and bandwidth resource for content delivery. As the decline of storage cost, most ordinary users' devices have been equipped with powerful storage with enormous capacity that usually cannot be fully occupied by users' own applications. Similarly, bandwidth resource is also cannot be completely saturated. Therefore, by paying users some rewards, they can be motivated to contribute their excessive resources to our system acting as the role of edge servers (but regarded as mini-servers here). Although the capacity of a single mini-server is nominal, we can easily recruit more mini-servers such that the total capacity can match the demand. Compared with traditional CDN systems which invest a lump sum of money in building and maintaining edge servers, the crowdsourcing-based system is more efficient with lower cost by saving the investment in both hardware and bandwidth. According to the literature [6], Thunder Crystal as a typical crowdsourcing-based CDNs can achieve about half price of traditional CDN services [6] because of its cheaper price to collect users' resources.

Despite above mentioned advantages, crowdsourcing-based CDNs confront the challenge that the bandwidth resource supplied by users' end devices is not as stable as that of edge servers in traditional CDNs because only users' residual bandwidth deducting the bandwidth consumed by other applications is available for content delivery. That means the uploading capacity of a mini-server is a variable depending on the user' status. The unstable bandwidth supply could cause severe bottleneck for content delivery services, especially for video delivery services, because it is possible that a user replicating some key segments of a file could suffer network congestion at any time. In addition, it is very difficult predict the bandwidth variation and make replication adjustment accordingly in time unlesswe can predict how much bandwidth will be used by an

This work was supported by the National Key R&D Program of China under Grant 2016YFB0201900, the National Natural Science Foundation of China under Grant 61572538, the Fundamental Research Funds for the Central Universities under Grant 17LGJC23 and the Australian Research Council under Discovery Projects DP150103658. (*Corresponding author: Di Wu)

Yipeng Zhou is with the School of Data and Computer Science with Sun Yat-sen University and Institute for Telecommunications Research with University of South Australia, email: yipeng.job@gmail.com; Terence H. Chan and Siu Wai Ho are with the Institute for Telecommunications Research with University of South Australia, email: hlchan6@gmail.com, siuwai.ho@unisa.edu.au; Guoqiao Ye and Di Wu are with School of Data and Computer Science with Sun Yat-sen University, email: yegq3@mail2.sysu.edu.cn, wudi27@mail.sysu.edu.cn

end user.

To overcome the drawback of unstable bandwidth supply, we propose to cache coded content on mini-servers such that they can complement each other when some of them suffer from drained network capacity . A probabilistic model is created to theoretically derive the performance gap between the schemes w or w/o coded content for replication. We prove that the performance gap improved by the coded scheme is more significant if less content is replicated by each mini-server or fewer segments a file is split into.

It is important to mention that the crowdsourcing-based system is fundamentally different from the peer-to-peer (P2P) file delivery system, in which peers help each other by sharing the content they are downloading [7], [8]. However, in crowdsourcing-based systems, though mini-servers are also recruited from ordinary Internet users, the content they are serving is independent with the content they are downloading. It means a central server needs to continuously push content to them for replication and they just serve user requests with best efforts. Rewards will be paid to mini-servers based on their upload traffic. In fact, a mini-server should be regarded as an edge server.

The rest content is organized as follows. The latest related work is elaborated in Sec. II. The background of the crowdsourcing-based content delivery system and our model is introduced in Sec. III. Performance analysis is conducted with static networking in Sec. IV and dynamic networking in Sec. V separately. Simulation is executed in Sec. VI to validate our model before our work is concluded in Sec. VII.

II. Related Work

Internet content delivery is always a challenging research problem because of the fast growing Internet traffic, especially the video traffic [9]. The system architecture for content delivery has been explored by numerous works to adapt with the rapid growth of user demands. For example, the P2P system is firstly designed by organizing users downloading the same content to share with each other such that the delivery cost can be minimized [10]–[12]. However, it is difficult to provide high quality delivery service. Cloud-based video streaming is then proposed by provisioning sufficient resources for user demands though it is a pricey service [13], [14]. The CDN system deploying a plenty number of edge servers around end users is one of the most prevalent and efficient architectures for content delivery, yet it still suffers from the surging management and delivery cost by maintaining a large-scale system [3]. Therefore, P2P assisted CDN system, i.e., hybrid system, is developed to balance the cost and quality of the content delivery service [15]. Different from above systems, the crowdsourcing-based CDN is an alternative solution that is efficient to control the operation cost of the CDN system by recruiting ordinary users as mini-servers. Rewards, e.g., cash, are paid back to recruited users to motive them to keep contributing resources [6], [16], [17]. In comparison with traditional CDN systems, both hardware cost and bandwidth cost can be significantly reduced. Our work introduces the coded scheme into such crowdsourcing-based systems, and explore the benefits.

The coding technique has been extensively applied for Internet content replication and delivery. For example, it has been proved that networking coding can improve the throughput for Internet content IP multicast if there exists bottleneck links in the network [18]. However, for application layer multicast, such as peer-to-peer content delivery system, or any systems without significant bottleneck links it is difficult to improve the system performance by simply adopting network coding for content transmission [19], [20]. Whereas, the network coding is helpful if the replication cost is involved. In particular, the network coding technique can be applied to encode the video content replicated on peers in P2P VoD systems. For example, Liu et al. [21] proposed a scheme to split each video file into multiple substreams. Each substream is encoded and replicated on each peer such that we can use less space to replicate each video. In this way, more videos can be pushed out to peers for replication to reduce replication cost hence improve streaming quality. Zhou et al. further extended the encoding strategies for video replication in P2P VoD systems which can trades off encoding/decoding cost and replication cost [22]. In P2P live streaming systems, though video content cannot be replicated on peers in advance, we can still apply network coding to simplify the request scheduling strategies because it is not necessary for downloading users to discriminate peers providing different video pieces [23], [24]. The network coding can also be utilized in distributed storage systems by replicating encoded content on servers. Then, the recover cost given the failure of any server can be reduced significantly [25]. Wu et al. [26] further investigated the application of network coding in content centric networks that not only improves caching efficiency but also protects user privacy.

III. Background

A. System Architecture

A crowdsourcing-based CDN system operates in a similar way as a traditional CDN except that edge servers are replaced by mini-servers. Edge servers are deployed by CDNs around end users that are capable to cache certain content and respond user requests. CDNs also need to keep updating edge servers with the latest content. In contrast, mini-servers are these devices crowdsourced from ordinary Internet users, which should be also updated continuously. To differentiate them, we use the term “mini-server" for crowdsourced CDNs.

Fig. 1 is an illustration of the crowdsourcing-based CDN system with a single file. We assume there exist a set of centralized file servers continuously pushing the latest (encoded or original) content to each mini-server. User requests are directed (by tracker servers) to corresponding mini-servers for data retrieval. A tracker server keeps track of how the file content has been replicated by each mini-server. Tracker servers can also manage and balance the load to avoid overloading any mini-servers. Under the crowdsource framework, mini-servers will be rewarded for their services, for example based on the upload traffic catering for the user requests [16], [27].

For replication, each file (e.g., a video) will be split into multiple segments (usually with the same size for ease of

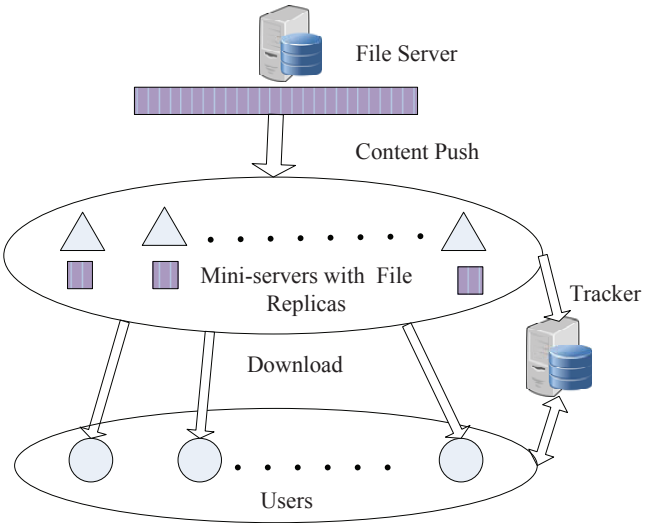


Fig. 1: Illustration of the system architecture with a single file.

management). Each mini-server can then replicate a subset pushed from centralized file servers prior to the start of file delivery[1]. Instead of simply replicating original segments, an alternative approach is to use coding. Central file servers can generate coded segments for each file and push them to mini-servers for replication. In this case, the original file content can be recovered as long as a sufficient number of coded segments have been retrieved by users. Note that the decoding operation needs to consume some additional computing resources, which however is acceptable for end users even if they are using battery constrained movie devices according to the study of previous works [28], [29].

During the retrieval process, a user can simultaneously download content from multiple mini-servers to maximize the retrieval efficiency. Similarly, mini-servers can equally split their uploading bandwidth among different users as well. In this paper, it is assumed that mini-servers are well-rewarded that they will upload content to users at maximum uploading rate (which may vary over time). To preclude the influence of request scheduling, we presume that an efficient scheduler has been deployed such that a mini-server will not starve for a moment as long as there exists any user needing content replicated on it. We will compare and evaluate the performances of crowdsourcing-based CDNs using coded or uncoded segments.

Discussion: Note that the practical system is more complicated due to the existence of a vast number of different files. File popularity or even segment popularity should be considered by replication algorithms to determine the times each file or segment replicated by mini-servers according to the design principles of both traditional CDNs and crowdsourcing-based CDNs [17], [30], which can significantly affect downloading performance. To exclude its influence and focus on evaluating the improvement achieved by coded replication, we investigate the case with a particular file. However, the improvement achieved using coded scheme will persist even if there are multiple different files replicated based on popularity distribution.

[1]Similar to the operation of traditional CDNs, a file's replicas can be adjusted anytime if necessary, but the majority are pushed to mini-servers in advance.

B. Comparison with Traditional CDNs

We need to illustrate the main differences between the mechanisms of traditional CDNs and crowdsourcing-based CDNs, since through which we can better understand how coded scheme benefits crowdsourcing-based CDNs.

Firstly, an edge server of the traditional CDNs is usually equipped with much more powerful storage and bandwidth resources which are usually sufficient to support thousands of users simultaneously. Thus, a particular user only needs to download content from a single edge server. If the contacted edge server is fully occupied, the user request will be redirected to another edge server with sufficient free resources. In contrast, the uploading capacity of mini-servers is so nominal that a user need to download the file content from multiple mini-servers concurrently [30], [31].

Secondly, the population of mini-servers is much larger than the population of edge server. It is costly to replicate complete file copies on mini-servers. A reasonable solution is to replicate partial content on each mini-server. In contrast, the cost is only moderate to completely replicate a number of different file copies on an edge server.

Thirdly, it is risky to replicate complete files on mini-servers that may be retrieved and abused by owners of mini-servers without the permission of content owners [30], [31].

In summary, it is a common scenario that a user simultaneously download content from multiple mini-servers each of which only replicate partial content in crowdsourcing-based CDNs. Thus, we can explore the benefits of coded scheme with this new scenario.

C. Symbols

We assume that there are n users served by m mini-servers. We will focus on a specific file and the statistics of the time required to distribute a content file (of size F symbols) to the users. For fair comparison, each mini-server will store the same amount of content, no matter whether the replication scheme is coded or uncoded. In the case of uncoded scheme, we assume that the file is split into w segments and each mini-server only replicates l segments where $l \neq w$. To balance the delivery capacity, each segment is replicated by $\frac{ml}{w}$ times by all mini-servers. A user can recover the file as long as all the w segments are completely retrieved. In the coded case, we will assume that a fraction of $x = \frac{l}{w}$ of the file will be stored in each mini-server. A user can recover the file as long as it has downloaded w coded segments in total from all the mini-servers, based on the assumption that all coded segments are independent (e.g., when MDS codes are used).

To simplify analysis, we first focus on *static scenario* by assuming that the uploading rate of each mini-server i (denoted by u_i) is a constant in each downloading session. Based on static results, we will later extend to the dynamic scenario. Without loss of generality, we assume that $u_1 \geq u_2 \geq \ldots u_m$. In the dynamic scenario, the uploading rate of a mini-server

i is a random variable U_i, which are assumed to be discrete, identically and independently distributed (i.i.d). Each U_i will take one of the K possible values from the set $\mathcal{U}_K \triangleq \{u'_1, u'_2, \ldots, u'_K\}$. Its probability mass function (PMF) and cumulative distribution function (CDF) are f_U and F_U respectively. In other words, $f_U(u'_k) = Pr(U_i = u'_k)$ and $F_U(u'_k) = Pr(U_i \leq u'_k)$.

For convenience, we list all major notations used in the paper in Table I.

TABLE I: The list of major notations.

Notation	Description
n	The number of total users.
m	The number of total mini-servers.
F	The size of the file to be delivered.
w	The number of segments the file has.
l	The number of segments replicated by each mini-server.
x	The fraction of the file replicated by each mini-server.
u_i	The uploading rate of mini-server i for the static scenario.
u'_i	The uploading rate of the i^{th} state for the dynamic scenario.
U_i	The random variable indicating the uploading rate of mini-server i for dynamic scenario.
T	The file downloading time for static scenario.
$E[T]$	The expected file downloading time for dynamic scenario.

D. Metrics

In this paper, we will use *file downloading time* as the metric to evaluate the system performance. It is defined as the total time required for all users to completely download the file. At the beginning of each session, all n users start to download the file at the same time. Let T_c and T_n respectively denote the file downloading time required for all users to complete their retrieval, via using coding or not. Our aim is to analyze T_c and T_n in static scenario, and characterize the performance gain due to coding, i.e., $E[T_n] - E[T_c]$, in dynamic scenario.

IV. Analysis of Static Scenario

This section focuses on the analysis with static networking, which serves as the fundamental basis for the analysis with dynamic networking.

A. Analysis for Coded Scheme

As mini-server i stores only encoded x fraction of the content file, it can upload at most $\min\{u_i T_c, nxF\}$ coded content in total to all users during the downloading session of time T_c. Therefore, the time T_c satisfies the following equation:

$$nF = \sum_{i=1}^{i=m} \min\{u_i T_c, nxF\}. \quad (1)$$

Here, we assume that all mini-servers can exhaustively use their uploading resource for content delivery until their content have been acquired by all users or file delivery process is over. The proof of this equation is quite straightforward and thus omitted here.

Proposition *1 (Lower bound):* The file downloading time T_c achieved by replicating encoded content on mini-servers given in Eq. (1) is the lower bound of the file downloading time for the scheme replicating plain content.

Proof: The proof is straightforward. For uncoded scheme, each mini-server at most can upload its l segments to n users. Let T_n be the time users need to complete file downloading. Then, the size of content uploaded by mini-server i denoted by A_i is $A_i \leq \min\{u_i T_c, nxF\}$, implying that the amount of content uploaded by each mini-server with the uncoded scheme is no more than the amount of content uploaded by each mini-server with the coded content. Hence, $T_n \geq T_c$. ■

In principle, for uncoded scheme, the bottleneck of the file downloading is from the segment with the least delivery capacity. Therefore, we should try to allocate the delivery capacity to each segment as even as possible. If a segment's delivery is completed prior to other segments. Then, we can try to reduce the file downloading time if the segment is delayed a little bit by allocating less delivery capacity such that the delivery of other segments is accelerated with the saved capacity. However, it is very difficult to perfectly allocate capacity evenly among segments for the uncoded scheme, which is a knapsack problem in static networking, not to mention the dynamic networking. Whereas, by replicating coded content, there is no need to discriminate different segments such that the perfect scheduling can be achieved easily. That is the underling principle to derive that $T_c \leq T_n$.

Based on Eq. (1), it seems that the expression of T_c is a complicated function with parameters $u_1, u_2, \ldots, u_m$ and x. In fact, for given $u_1, \ldots, u_m$, the function T_c can be expressed as a piecewise linear function with x. To conveniently discuss how x affects downloading performance, We define a function $\tilde{T}(x)$ as below.

Definition 1: For fixed $u_1, \ldots, u_m$ and F, let $\tilde{T}(x)$ be defined the solution of the following equation

$$nF = \sum_{i=1}^{i=m} \min\left\{u_i \tilde{T}(x), nxF\right\}. \quad (2)$$

If there is no a pair of mini-servers with the same uploading rate, we have:

Proposition *2:* [Piecewise linearity] $\tilde{T}(x)$ is a piecewise linear decreasing function. In particular,

$$\tilde{T}(x) = \frac{nF(1 - ix)}{\sum_{i+1}^{m} u_j} \text{ if } x_{i+1} < x \leq x_i.$$

Here, we define $x_0 = 1$, $x_m = 0$ and

$$x_i = \frac{u_i}{\sum_{i+1}^{m} u_j + iu_i}$$

for $1 \leq i \leq m-1$.

Proof: Please refer to the detailed proof in Appendix. ■

Discussion: According to Proposition 2, in each interval $(x_{i+1}, x_i]$, $\tilde{T}(x)$ is a straight line with slope $-\frac{inF}{\sum_{i+1}^{m} u_j}$, which decreases with i. Thus, $\tilde{T}(x)$ is a convex function. If x approaches 1, the downloading performance is very good. If x approaches 0, $\tilde{T}(x)$ approaches infinity. In fact, when x is close to 1, the performances of both coded and uncoded schemes are very good; otherwise when x is close to 0, both schemes perform very bad implying a very small performance gap between two schemes. Therefore, the coded scheme can

achieve significant performance gain if the value of x is neither too small nor too large.

Now, we extend the Proposition 2 by considering more practical factors.

Corollary *1:* The number of mini-servers with uploading traffic nxF will never exceed $\lfloor \frac{1}{x} \rfloor$.

The proof of the corollary is straightforward by extending Proposition 2. If there are $\lfloor \frac{1}{x} \rfloor + 1$ mini-servers uploading more than nxF content each, then the total delivered content amount already exceeds nF, implying that the file delivery process is over. In large-scale systems, the value $\lfloor \frac{1}{x} \rfloor$ is usually much smaller than m. This means that the number of mini-servers which quit file delivery before T_c is likely to be very small compared with the population of mini-servers in real world.

In practice, it is common that multiple mini-servers may have the same uploading rate because the mini-server population m should be much more than the number of uploading states K. This is a reasonable assumption for large-scale systems with tens of thousands of mini-servers. Recall the sample space of uploading rate is $u'_1, u'_2, \ldots, u'_K$. Thus, if $m \gg K$, it implies that the number of mini-servers with uploading rate u'_k must be larger than 0. To derive T_c in this case, we need to enumerate the number of mini-servers with uploading rates larger than u'_k who will abort file delivery before T_c or the probability that i mini-servers abort file delivery with $u_i = u'_k$. Since the mini-servers with the same uploading rate must abort or not together, the event that i mini-servers abort implies that $u_i = u'_k$, $u_{i+1} = u'_{k+1}$ for some k. Thus, given that the uploading rates have K possible values, we have:

Proposition *3:* [Piecewise linearity with K states] If i mini-servers abort file downloading and $u_i = u'_k$ and $u_{i+1} = u'_{k+1}$ with $u'_1 > u'_2 > \cdots > u'_K$, then $\tilde{T}_c(x) = \frac{nF(1-ix)}{\sum_{i+1}^{m} u_j}$ and $x_{k+1} < x \leq x_k$. Here, we define $x_0 = 1$, $x_K = 0$ and $x_k = \frac{u'_k}{\sum_{i+1}^{m} u_j + i u'_k}$ for $1 \leq k \leq K-1$.

This proposition is just a simple extension of proposition 2. The detailed proof is omitted. Proposition 3 is essential for us to analyze the content delivery performance in dynamic networking by reducing the computing complexity which will be presented in detail in the next section.

Similarly, for fixed x, the file delivery time can be denoted as a function $\tilde{T}_c(u_1, \ldots, u_m)$ with m parameters. In fact, this is the problem to be considered in the dynamic scenario which will be presented in the next section.

B. Analysis for Uncoded Scheme

Analysis of the uncoded scheme (where file content is simply divided into uncoded segments and got replicated at the mini-servers) is quite complicated. Results will depend on the networking state, i.e., $u_1, \ldots, u_m$, the values of l and w, and how exactly the content is replicated at mini-servers. For some simple choices of $u_1, \ldots, u_m$ and x, it may not be difficult to find an optimal replication strategy. For example, suppose $l = 1$, $w = 2$, and there are 100 fast mini-servers with uploading rate 1.0 Mbps and 100 slow mini-servers with 0.1 Mbps. The optimal replication strategy is to store each segment on 50 fast mini-servers and 50 slow mini-servers. In this case, $T_n = \tilde{T}(0.5)$ is the best performance. However, in other case where $l = 3$ and $w = 7$, it becomes difficult to determine how to replicate the uncoded segments at the mini-servers for optimal performance. More importantly, the real system is dynamic and hence its uploading rates will vary over time. A currently optimal replication strategy may no longer be optimal in the next instance. Thus, it can be very challenging to design a replication strategy that is robust and can automatically adapt with varying networking conditions. As a result, it will be of great interest to understand the performance of an uncoded system both in the best case and the worst case. In fact, the best case is the asymptotic case by letting l and w approach infinity proportionally, while the worst case is achieved when $l = 1$.

1) Asymptotic Analysis: It is sharp to compare the analysis of the uncoded system with the coded system. Thus, we define the following three events for each mini-server:

- E_1: The mini-server quits file delivery due to poor segment replication.
- E_2: The mini-server quits file delivery when its upload traffic reach xnF.
- E_3: The mini-server quits when the file delivery process completes.

For any mini-server, it quits file delivery if and only if any of the three events occurs. From Eq. (1) for coded system, the best delivery performance will be achieved if all mini-servers only leave due to either E_2 or E_3. For the uncoded case, E_1 is the event causing inefficiency and should be minimized. Thus, we should minimize the probability of E_1 so as to maximize the occurrence odds of E_2 or E_3 to optimize file delivery performance.

We introduce the definition of downloading stage to analyze the downloading process of uncoded content with smaller granularity. The whole file delivery process is partitioned into w stages based on the number of segments which have been delivered to all users. In other words, stage d is defined as the time when any d segments are delivered to all users to the time when any $d+1$ segments are delivered to all users. Ideally, we hope a mini-server will not quit file delivery before its uploaded traffic reaches xnF or the delivery process is over, which is the situation for coded systems. Otherwise, the mini-server is regarded as an early aborting one which defers the file delivery process. We compare the performance of two schemes by comparing mini-servers' early aborting chance at each stage.

We define $Q_i(w, l, d)$ with $d < w$ as the probability that E_1 occurs for the mini-server i at the end of stage d. As we have shown in Eq. (1) for coded scheme, a mini-server will not quit file delivery before T_c unless E_2 occurs. Whereas, for uncoded scheme, there exists the odds that a mini-server quits the file delivery process due to poor replication, which is denoted by $Q_i(w, l, d)$. Higher $Q_i(w, l, d)$ implies more wastage and worse performance.

Apparently, $Q_i(w, l, d) = 0$ if $d < l$ since a mini-server with l replicated segments can always have a needed segment for

users. For $l \leq d < w$, we have:

$$Q_i(w,l,d) = \frac{\binom{d}{l}}{\binom{w}{l}}, \quad (3)$$

by assuming a random balanced segment placement is adopted which replicates each segment $\frac{ml}{w}$ times.

Lemma *1:* $Q_i(w,l,d)$ is a monotonic increasing function with d for $l \leq d < w$., and $\frac{Q_i(w,l,d)}{Q_i(w,l,d+1)} \leq 1-x$.

Proof: The proof is straightforward since $\frac{Q_i(w,l,d)}{Q_i(w,l,d+1)} = \frac{\binom{d}{l}}{\binom{d+1}{l}} = \frac{d-l+1}{d+1} \leq \frac{w-l}{w} = 1-x < 1$. ∎

Lemma *2:* For any small number $\epsilon > 0$, we can always find an integer $z = O(\frac{\ln \epsilon}{\ln 1-x})$ such that $Q_i(w,l,d) < \epsilon$ for $d < w-z$.

Proof: Given Lemma 1 that $\frac{Q_i(w,l,d)}{Q_i(w,l,d+1)} \leq \frac{w-l}{w}$, $Q_i(w,l,d) < \epsilon$ as long as $w - d > \frac{\ln \epsilon}{\ln 1-x}$. By letting $z = \frac{\ln \epsilon}{\ln 1-x} + C = O(\frac{\ln \epsilon}{\ln 1-x})$ where C is a constant, based on Lemma 1 we have $Q_i(w,l,d) < \epsilon$ for $d < w - z$. ∎

Proposition *4:* [Asymptotic Performance for Uncoded Scheme] Suppose random balanced replication strategy is used where each uncoded segment will be replicated $\frac{nl}{w}$ times and randomly pushed to mini-servers. Then T_n can be arbitrarily close to T_c, i.e., $T_n - T_c < \epsilon$ for any small number ϵ, as long as w is sufficiently large with fixed $x = \frac{l}{w}$.

Proof: The detailed proof is presented in appendix. ∎

We briefly discuss the insight of this result here. Fixed x means that l will also increase proportionally as we increase w. Although downloading performance can be improved by increasing w and l proportionally (which can be achieved by decreasing segment size) with fixed replication cost $x = \frac{l}{w}$, it will increase the load for replication management and download scheduling severely. In a content delivery system, a tracker server needs to record what segments have been replicated by each mini-server so that a user can be noticed in time to download the right segments from corresponding mini-servers. The cost to maintain the tracker server is at least proportional with the population of segments to be managed. In addition, if the segment size is very small, users have to switch their downloading connections among many different mini-servers frequently, which can also dramatically degrade downloading performance.

2) Worst Case of Uncoded System: Now, we turn to investigate the downloading performance under a special case with $l = 1$ because of the following two reasons. On one hand, practical system operators prefer to replicate only one segment of a particular file on each server due to its simplicity for management. On the other hand, we can derive the worst bound of downloading performance.

If $l = 1$, all mini-servers form into w clusters and each cluster just serves a particular segment. The performance bottleneck is determined by the segment with least capacity. Let T_n^L denote the maximum value of T_n, then

$$T_n^L = \frac{nF}{w \sum_{\frac{m}{w}+1}^{m} u_j}, \quad (4)$$

where $u_1 > \cdots > u_m$. Here $\frac{nF}{w}$ is the amount of content to be delivered by each cluster. The worst case is achieved if a particular segment is replicated by the smallest $\frac{m}{w}$ mini-servers. Thus, the worst bound of file downloading time is given by Eq. (4). In fact, in static networking the downloading performance of the uncoded scheme with an arbitrary replication strategy is very complicated because it depends on how these segments are placed on mini-servers which is out of our control. Fortunately, in the dynamic networking we can derive the expected time.

V. Analysis of Dynamic Scenario

With the analysis results of the static scenario, we can move on to analyze the dynamic scenario in this section.

This section is essential for crowdsourcing-based CDN systems. As we have emphasized in Sec. III, in practice, the bandwidth resource collected from ordinary users will fluctuate significantly because the bandwidth must be shared by users' different applications. Thus, it is unreasonable to assume the uploading rate of a mini-server is known in advance or fixed. Therefore, we need to investigate this problem by assuming that the uploading rates of mini-servers independently varies following an identical distribution.

In dynamic networking, the uploading rate of each mini-server is assumed to be a discrete random variable, which is defined as $f_U(u'_k) = Pr(U_i = u'_k)$ and $\sum_{k=1}^{K} f_U(u'_k) = 1$. We further assume $u'_1 > u'_2 > \cdots > u'_K$. The uploading rate of each mini-server changes independently, but it is unchanged during each file downloading session for simplicity. Our objective is to derive the expected file downloading time for both schemes. We let $E[T_c]$ and $E[T_n]$ denote the expected file downloading time for coded and uncoded schemes respectively.

A. Analysis for Coded Scheme

For any given x, the most straightforward method to compute $E[T_c]$ is to enumerate all possible network states, i.e., $E[T_c] = \tilde{T}_c(u_1,\ldots,u_m) \times Pr(u_1,\ldots,u_m)$. By abusing the notation a little bit, $\tilde{T}_c(u_1,\ldots,u_m)$ represents the downloading time as a function of $u_1,\ldots,u_m$. The computation load of the expected time is too heavy even if we restrict that there are only K uploading states since the computing complexity is of exponential magnitude.

Fortunately, the complexity can be reduced to $O(mK)$ by using Proposition 3. For any given x, the file delivery time will be $\frac{nF(1-ix)}{\sum_{i+1}^{m} u_j}$ given that i mini-servers depart before T_c, which implies $x_{k+1} < x \leq x_k$. Here, $x_k = \frac{u'_k}{\sum_{i+1}^{m} u_j + iu'_k}$, $x_0 = 1$ and $x_K = 0$. Thus, the expected file downloading time is:

$$\begin{aligned} E[T_c] = & \sum_{k=1}^{K}\sum_{i=0}^{m} \frac{nF(1-ix)}{\sum_{i+1}^{m} u_j} \times \\ & Pr\left(x_{k+1} < x \leq x_k | u_i = u'_k, u_{i+1} = u'_{K+1}\right) \times \\ & Pr(u_i = u'_k, u_{i+1} = u'_{K+1}). \end{aligned} \quad (5)$$

In Eq. (5), we enumerate the possibility that i mini-servers will abort in advance with the condition that $u_i = u'_k$ and $u_{i+1} = u'_{k+1}$. For each possibility, there is a product of three terms, which are file downloading time, the probability to satisfy the constraint of x and the probability to meet the uploading rate distribution respectively.

$Pr\left(x_{k+1} < x \leq x_k | u_i = u'_k, u_{i+1} = u'_{K+1}\right)$ can be computed based on Proposition 3. $x_{k+1} < x \leq x_k$ given that $u_i = u'_k, u_{i+1} = u'_{K+1}$ is equivalent to $\sum_{i+1}^{m} U_j \leq \frac{u'_k}{x} - iu'_k$ and $\sum_{i+1}^{m} U_j > \frac{u'_{k+1}}{x} - iu'_{k+1}$. Thus,

$$\begin{aligned} & Pr\left(x_{k+1} < x \leq x_k | u_i = u'_k, u_{i+1} = u'_{K+1}\right) \\ = \; & Pr\left(\frac{u'_{k+1}}{x} - iu'_{k+1} < \sum_{i+1}^{m} U_j \leq \frac{u'_k}{x} - iu'_k\right) \\ = \; & Pr\left(\frac{u'_{k+1}}{(m-i)x} - \frac{iu'_{k+1}}{m-i} < \frac{\sum_{i+1}^{m} U_j}{m-i} \leq \frac{u'_k}{(m-i)x} - \frac{iu'_k}{m-i}\right). \end{aligned}$$

Since mini-servers change their uploading states independently and identically, approximately $\frac{\sum_{i+1}^{m} U_j}{m-i}$ follows Gaussian distribution for large systems with a sufficient number of mini-servers.

Similarly, the third term can be derived as

$$\begin{aligned} & Pr(u_i = u'_k, u_{i+1} = u'_{K+1}) \\ = \; & \binom{m}{i} Pr^i(U \geq u'_k) Pr^{m-i}(u \leq u'_{k+1}). \end{aligned} \tag{6}$$

Together with above derivations, we can approximately derive $E[T_c]$ as long as we know the probability mass function of the uploading state, i.e., $Pr(u'_k)$.

Discussion: In the computing of the expected downloading time, we enumerate K states in Eq. (5) by assuming that $K \ll m$. Otherwise, according to Proposition 2, we need to enumerate m possibilities in the worst case. Then, for large scale system, the computing load $O(m^2)$ could be still very heavy. From the implication of this computation, we can propose to reduce computation time for the case with too many network states. More specifically, we can group all network states or even partition a continuously varying network state into a limited number of K states though a little inaccuracy could be incurred. For example, for the case that the distribution of uploading rate follows a probability density function, we can reduce the computation complexity by partitioning the range of uploading rates into K intervals. The probability for the uploading rate to fall into each interval is just the cumulative probability of that interval. Then, we can approximately generate a PMF for uploading rates which can be efficiently used to compute the expected file delivery time.

B. Analysis for Uncoded Scheme

For analysis of the uncoded scheme, we again first prove that the performance is asymptotically optimal as m and n approach infinity before we derive $E[T_n]$ for the special case with $l = 1$.

Proposition *5:* In dynamic networking, as m and n approach infinity with fixed $\frac{m}{n}$, then for any small number ϵ, we always have $\lim_{n\to+\infty} Pr\left(|T_n - T_c| > \epsilon\right) = 0$.

Proof: This proof is straightforward by using the law of large number. We just briefly discuss the proof. Based on Proposition 4, the worst performance is achieved for the uncoded scheme when $l = 1$. We prove this proposition by setting $l = 1$. If $l = 1$, all mini-servers can be classified into w groups, and members within the same group replicate the same file segment. Define C_j be the random variable denoting the network capacity to distribute segment j, i.e., the aggregated uploading rate of mini-servers in group j. Its PMF and CDF are denoted by $g(C_j)$ and $G(C_j)$ respectively. Then, based on the law of large number, $\lim_{n\to+\infty} Pr\left(|\frac{wC_j}{m} - E[U]| > \epsilon'\right) = 0$ for any positive number ϵ'. On the other hand, $C_{min} = \min\{C_1, \ldots, C_w\}$ is the bottleneck of the whole system determining the file downloading time. Thus, as n approaches infinity, C_{min} will converge to the expected value $E[C_j]$ in probability implying the hold of proposition 5. ■

Discussion: In theory, by simply replicating a single segment on each mini-server, the file downloading performance converges to the optimal performance with the increase of system scale. However, in practical systems, it is very difficult to achieve such perfect performance. This proposition requires that the uploading rates of different mini-servers must be i.i.d. variables. Firstly, it is likely that the distributions of mini-servers' uploading rates are distinct because they purchase different Internet access plans resulting in heterogeneous distributions, which should be factored in to balance the delivery capacity of each segment. Secondly, the uploading rates of some mini-servers may be highly correlated. For example, mini-servers within the same enterprise will be affected by the surplus bandwidth of the enterprise simultaneously. Such correlated bandwidth variations will increase the variances of the capacity to distribute file segments. Thirdly, the churn of mini-servers [2] also increase the variance of capacity distribution. But, in contrary, by simply replicating coded content, we can achieve the optimal performance without the need to consider all above complicated factors.

The case with $l = 1$ is of particular importance for practical systems because of its simplicity for implementation and maintenance. For instance, in the real system Thunder Crystal [16], l is set as 1. Thus, we further discuss this case in the following context. Let u and σ denote the mean and variance of the uploading rate U. Based on the law of large number, we can further derive that the expected file downloading time for the uncoded scheme is:

$$E\left[T_n\right] = \int_0^{+\infty} \frac{nF}{y} g(y) \left(1 - G(y)\right)^{w-1} dy, \tag{7}$$

where $g(y) = \frac{\sqrt{w}}{\sqrt{m}\sigma}\phi\left(\frac{wy-mu}{\sqrt{mw}\sigma}\right)$, and $G(y) = \Phi\left(\frac{wy-mu}{\sqrt{mw}\sigma}\right)$. ϕ and Φ are the PDF and CDF of the standard Normal distribution respectively. Intuitively speaking, if downloading time is larger than t, it implies that the capacity of C_{min} is less than $\frac{nF}{wt}$, based on which we derive the expected value. For detailed proof of Eq. (7), please refer to appendix.

Although the result looks complicated, it is not difficult to solve it numerically.

Discussion: All above derivations are based on the assumption that the distribution of the uploading rate is identical for all mini-servers. In fact, it is highly possible that uploading rates follow a number of different independent distributions. For the general case, the computation of the expected file downloading time for the coded scheme will be very difficult

[2]Mini-servers may leave the system and turn off their devices temporarily due to some reasons.

since the complexity to compute Eq. 6 is exponential if we have several different distributions for the uploading rate. Despite its complication, we explain its influence intuitively as follow. The distribution of uploading rates is more skewed if the uploading rates follow several different distributions, and the downloading performance will become worse for uncoded scheme because users cannot adaptively download more content from the faster mini-servers with larger capacity. In contrast, if coded content is replicated, users will automatically adjust download amount from mini-servers, thereby the vulnerability will be smaller for coded scheme and the gap between uncoded and coded schemes becomes wider.

VI. Performance Evaluation

In this section, simulation is conducted to compare the performance gap between the coded scheme and the uncoded scheme, and validate the accuracy of our theoretical model.

A. Simulation Settings

In all simulations, the size of the file to be delivered is $F = 1000$ Mb.

By default, the file is split into $w = 10$ segments and each mini-server replicates $l = 1$ segment, unless we explicitly state otherwise. we use a system with $m = 1000$ mini-servers and $n = 1000$ users for simulation. The uploading rate of each mini-server is either $u_l = 0.1Mbps$ or $u_h = 1Mbps$. In other words, $f_U(u_l) = 0.5$ and $f_U(u_h) = 0.5$. We just use the samples with 0.1 Mbps and 1.0 Mbps as an example for the simulation to illustrate our scheme. In fact, they can be set with any other values, and the performance improved with our scheme will persist. For static networking, we set that there are 500 mini-servers with 0.1 Mbs and 1.0 Mbps uploading rates respectively. For dynamic networking, the uploading rate changes after each file downloading session. To achieve reliable simulation performance and stable performance curves, the result of each point is the average value of 100 independently simulated file delivery sessions.

With the distribution of uploading states, i.e., $f_U(u_l) = f_U(u_h) = 0.5$, we can compute the theoretical expected file downloading time in dynamic networking using Eq. (5) and Eq. (7) for coded scheme and uncoded scheme separately.

Note that $S-Coding$ and $S-NoCoding$ represent the results generated by simulations; while $T-Coding$ and $T-NoCoding$ are results produced by theoretical models in the figures showing simulation results.

The purpose of our simulations is to provide system designers a way to quantify the transmission benefits by adopting coded content for replication. In practical systems, the efficiency of the coded/uncoded scheme is also very essential for the finally achieved performance gain. To name a few, how to split chunks, how to choose the file size and the implementation of specific encoding and decoding algorithms are all important aspects which should be thoroughly considered by system designers. However, it is more feasible to evaluate these aspects in a practical system instead of artificially designed simulation environment.

B. Simulation Results

We first use a small system with $m = 10$ mini-servers and $n = 10$ users to validate the result of Proposition 2. The uploading rate of mini-server i is $1.1 - 0.1i$ Mbps for $1 \le i \le 10$.

Fig. 2 is the simulation result of this fully connected small network. Here, we fix $l = 1$. By varying the value of x from $1/2$ to $1/10$, i.e., varying w from 2 to 10, we can observe that the coded scheme significantly outperforms the uncoded scheme, especially for the value of x in the range $(\frac{1}{8}, \frac{1}{6})$. The theoretical result obtained from Proposition 2 is also plotted in the figure which is almost overlapped with the simulation curve.

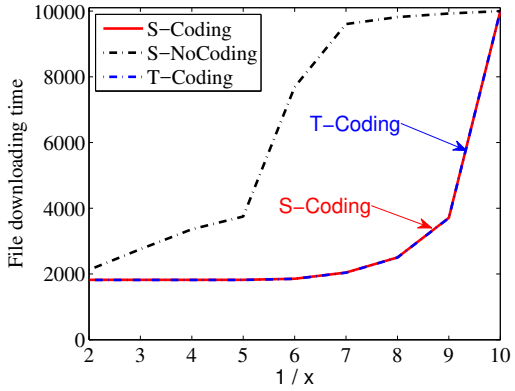


Fig. 2: Simulation result of a small system with static networking.

Fig. 3 is the simulation result of the system with $m = 1000$ and $n = 1000$ by varying the number of requests sent out by each user. The simulation serves the purpose to study how the network connectivity affects content delivery performance. The networking is static, i.e., there are 500 fast mini-servers with uploading rate at 1.0 Mbps and 500 slow mini-servers with uploading rate at 0.1 Mbps. In principle, the system performance should be better if each user can randomly send out more requests to different mini-servers resulting in a more balanced state. By varying the number of requests from 10 to 100, we find that the performance is quite stable as shown in Fig. 3, which indicates that the performance is only slightly affected by the network connectivity as long as the number of requests exceeds a certain threshold. In the following simulations, the number of requests of each user is fixed at 10.

In Fig. 4, we evaluate how the value x affects system performance in static networking by varying w from 100 to 1000 with fixed $l = 1$. As the result in Fig. 4 shows, the performance gap between the coded scheme and the uncoded scheme is not apparent when x is either very large or very small. But when $\frac{1}{x}$ is in between 400 and 700, the content delivery time could be twice as large as that of the coded scheme. Compared with Fig. 2, the x to achieve the maximum gap is much smaller in Fig. 4 because of the larger system scale. Note that the curve of uncoded scheme is a jagged curve. Along the curve, each peak is caused by the unbalanced

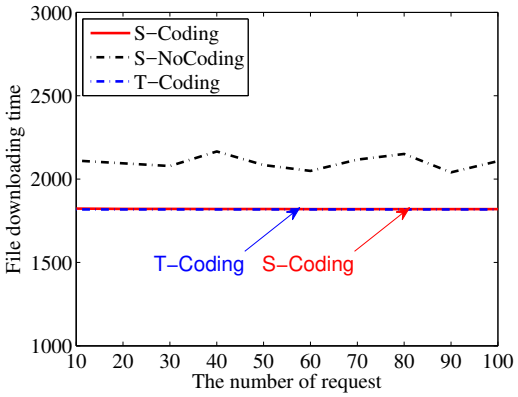


Fig. 3: Simulation result by varying the number of requests sent by each user from 10 to 100 in static networking.

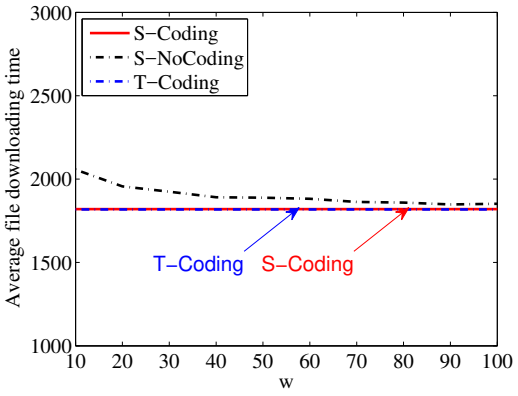


Fig. 5: Simulation result by increasing l from 1 to 10 and w from 10 to 100 with fixed $x = \frac{1}{10}$ in dynamic networking.

resource allocation because the segment replication is uneven when m cannot be exactly divided by w.

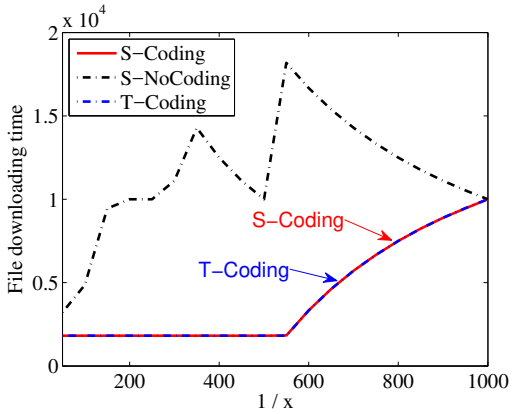


Fig. 4: Simulation result by decreasing x from $\frac{1}{100}$ to $\frac{1}{1000}$ in static networking.

The result presented Fig. 5 is achieved by increasing w and l proportionally with fixed x at $1/10$. As we have analyzed in Proposition 4, the content delivery performance of the uncoded scheme will asymptotically approach the performance of the coded scheme as w and l approach infinity proportionally. We validate this assertion by executing the simulation with w varying from 10 to 100 and l varying from 1 to 10. The networking is dynamic as each mini-server has $1/2$ probability to switch to either the fast state with 1.0 Mbps uploading rate or the slow state with 0.1 Mbps uploading rate at the beginning of each file delivery session. Each point of the simulation curve is the average value of 100 independently executed file downloading sessions. As we can see, the performance gap between two schemes gradually diminishes with the increase of w and l, which is consistent with the conclusion of Proposition 4.

The simulation presented in Fig. 6 is used to validate the theoretical formulas derived to compute the expected downloading time. In this simulation, we vary x from $1/10$ to $1/100$ for both schemes with w varying from 10 to 100. The theoretical expected delivery time is computed using Eq. (5) and Eq. (7). The networking is dynamic with the same setting with the last simulation. As we can see, the theoretical curves and the simulation curves are very close with negligible gaps for all cases, which indicates the accuracy of our theoretical model and the advantages of the coded scheme.

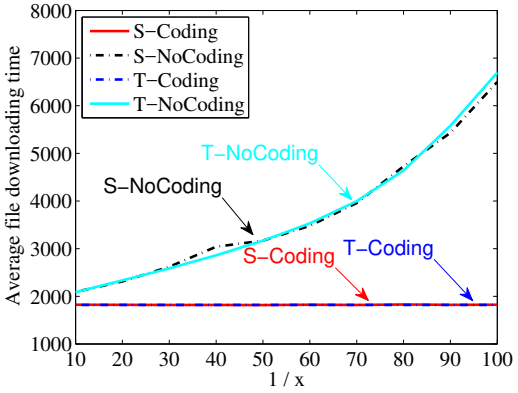


Fig. 6: Simulation result by decreasing x from $\frac{1}{10}$ to $\frac{1}{100}$ with w varying from 10 to 100 in dynamic networking.

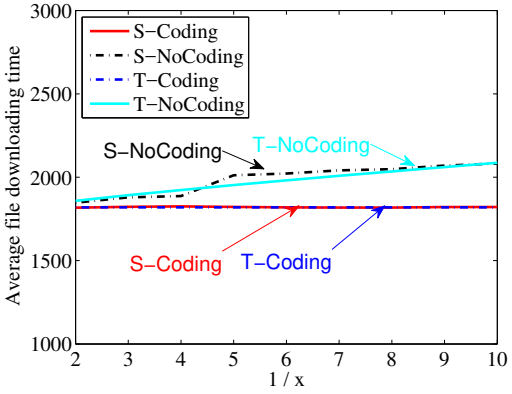


Fig. 7: Simulation result by fixing $l = 1$ and increasing w from 2 to 10 with 10 uploading states, i.e., $0.1, 0.2, \ldots, 1.0$ Mbps.

Fig. 7 is the simulation result with 10 uploading states, i.e.,

0.1, 0.2, . . . , 1.0 Mbps, in dynamic networking. The occurrence of each state for uploading rate is 1/10. In this simulation, we fix $l = 1$ and increase w from 2 to 10. The other settings are the same with the last simulation. Each case is simulated for 100 times to compute the average downloading time. Again, by comparing the performance of the two schemes, the coded scheme achieves better performance, especially for the cases with smaller x. The theoretical curves and simulation curves are very close, which further validates the accuracy of our model.

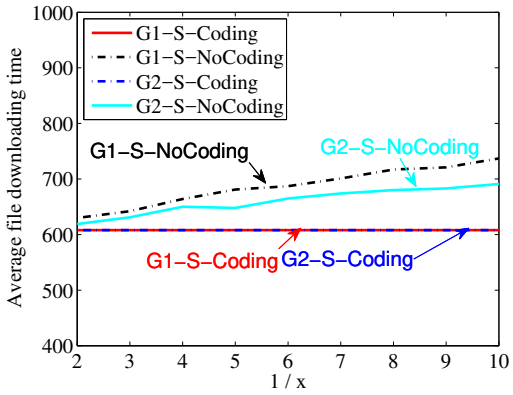


Fig. 8: Simulation result by fixing $l = 1$ and increasing w from 2 to 10 in heterogeneous case in which mini-servers have different distributions for uploading rates.

In our analysis, we assume that the distribution of uploading rates is identical for all mini-servers. In practice, it is common that they may have different distributions. To evaluate how it affects performance, we conduct a simulation with two distributions. More specifically, for the first distribution, there are 500 slow mini-servers with 0.1 Mbps with 0.5 probability and 1.0 Mbps with 0.5 probability. There are 500 mini-servers with 0.5 Mbps with 0.5 probability and 5.0 Mbps with 0.5 probability for the second distribution. This is regarded as $G1$ simulation. As a comparison, we set up another simulation, i.e., $G2$, with identical distribution for uploading rate. To have a fair comparison, we set up 1000 mini-servers with 0.3 Mbps with 0.5 probability and 3.0 Mbps with 0.5 probability so as to have the same total uploading rate with $G1$. We vary x from $\frac{1}{2}$ to $\frac{1}{10}$ by fixing $l = 1$. The simulation result is presented in Fig. 8, which shows that the performance for coded scheme is the best and almost the same for both simulations. Whereas, the performance of $G1$ with uncoded scheme is the worst since the distribution of uploading rates is more skewed. This simulation result is also consistent with our discussion in Sec. V.

VII. Conclusion

Recently, crowdsourcing-based systems have attained extensive study focusing on the incentive mechanism, architecture design, system measurement and privacy preservation. Differently, in this paper, we propose a scheme to cache coded content on mini-servers in such systems to deal with the challenge of unstable bandwidth resource crawled from ordinary Internet users. We attested the advantage of the coded scheme through both theoretical derivations and simulation experiment. Although the uncoded scheme can asymptotically achieve the same performance with the coded scheme, the overhead cost is too heavy and unbearable form real systems. In practice, the coded scheme can significantly outperform the uncoded scheme with limited resources and simple segment splitting. Extending our framework to support video streaming applications by replicating coded content will be our future work.

References

[1] M. Arantes, F. Figueiredo, and J. M. Almeida, "Understanding video-ad consumption on youtube: A measurement study on user behavior, popularity, and content properties," in *Proceedings of the 8th ACM Conference on Web Science*. ACM, 2016, pp. 25–34.

[2] A. Balachandran, V. Sekar, A. Akella, and S. Seshan, "Analyzing the potential benefits of cdn augmentation strategies for internet video workloads," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 43–56.

[3] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid cdn-p2p: why limelight needs its own red swoosh," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2008, pp. 75–80.

[4] "Xiaomi mi wifi wireless ac router," http://www.xiaomishop.com/128-original-xiaomi-mi-wifi-wireless-router.html.

[5] Wikipedia, "Bt home hub," 2014, [Online; accessed 21-Jul-2013]. [Online]. Available: \url{http://en.wikipedia.org/wiki/BT_Home_Hub}

[6] 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.

[7] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 2, pp. 566–579, 2013.

[8] P. Shah and J.-F. Paris, "Peer-to-peer multimedia streaming using bittorrent," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*. IEEE, 2007, pp. 340–347.

[9] M. Hazas, J. Morley, O. Bates, and A. Friday, "Are there limits to growth in data traffic?: on time use, data generation and speed," in *Proceedings of the Second Workshop on Computing within Limits*. ACM, 2016, p. 14.

[10] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *International Workshop on Peer-to-Peer Systems*. Springer, 2005, pp. 205–216.

[11] Y. Zhou, D.-M. Chiu, and J. Lui, "A simple model for chunk-scheduling strategies in p2p streaming," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 1, pp. 42–54, 2011.

[12] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "On replication algorithm in P2P vod," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 233–243, 2013. [Online]. Available: http://dx.doi.org/10.1109/TNET.2012.2196444

[13] 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.

[14] J. He, Y. Wen, J. Huang, and D. Wu, "On the cost–qoe tradeoff for cloud-based video streaming under amazon ec2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, 2014.

[15] C. Huang, J. Li, K. W. Ross *et al.*, "Peer-assisted vod: Making internet video distribution cheap." in *IPTPS*, 2007.

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

[17] 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.

[18] 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.

[19] 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.

[20] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan, "Can network coding help in p2p networks?" in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium on*. IEEE, 2006, pp. 1–5.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[26] Q. Wu, Z. Li, G. Tyson, S. Uhlig, M. A. Kaafar, and G. Xie, "Privacy-aware multipath video caching for content-centric networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2219–2230, 2016.

[27] Y. Zhou, L. Chen, M. Jing, Z. Ming, and D. M. Chiu, "Analyzing streaming performance in crowdsourcing-based video service systems," in *SZU Technical Report*, 2015. [Online]. Available: http://hpc.szu.edu.cn/ypzhou/zcjmc15tr.pdf

[28] F. H. Fitzek, M. V. Pedersen, J. Heide, and M. Médard, "Network coding: applications and implementations on mobile devices," in *Proceedings of the 5th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. ACM, 2010, pp. 83–87.

[29] J. Heide, M. V. Pedersen, F. H. Fitzek, and T. Larsen, "Network coding for mobile devices-systematic binary random rateless codes," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[30] A.-M. K. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, vol. 4, 2007.

[31] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

APPENDIX

Proof of Proposition 2:

As we have shown in Eq. (1), a mini-server may quit the file distribution task prior to T_c if its content has been distributed to all n users. Given $u_1 > \cdots > u_m$, mini-servers will quit with the sequence $1, 2, \ldots, m$. Let x_i be the value such that just i mini-servers quit prior to T_c. Then, we have the following two equations:

$$x_i F = u_i T_c(x_i), \quad (8)$$

$$nF - i x_i F n = T_c(x_i) \sum_{j=i+1}^{m} u_j, \quad (9)$$

for $0 < i < m$. The meaning of the first equation is that each mini-server who quits prior to T_c just uploads its content to n users. The remaining mini-servers will be in charge of distributing the remaining content. Based on the two equations, we get:

$$x_i = \frac{u_i}{\sum_{j=i+1}^{m} u_j + i u_i},$$

$$T_c(x_i) = \frac{nF}{\sum_{j=+1}^{m} u_j + i u_i},$$

for $0 < i < m$. By taking the boundary conditions into account, we let $x_0 = 1$ and $x_m = 0$. If $x_{i+1} < x \le x_i$, it means that i mini-servers will leave prior to T_c, and

$$T_c(x) \sum_{j=i+1}^{m} u_j = nF - ixFn.$$

Thus, $T_c(x) = \frac{nF(1-ix)}{\sum_{j=i+1}^{m} u_j}$. ∎

Proof of Proposition 4:

We prove this proposition by showing that the probability of E_1 can be arbitrarily small for all stages as long as w is sufficiently large.

More specifically, from the proof of Lemma 2, we have shown that for any small value ϵ', we always have $Q_i(w, l, d) < \epsilon'$ for all $1 \le d \le w - z$ where $z = O(\frac{\ln \epsilon}{\ln 1-x})$. Then, all stages can be divided into two groups with 1 to $w - z$ as the first group and the others as the second group. For the second group with w stages, with w approaches infinity, $\frac{z}{w} < \frac{1}{w} O(\frac{\ln \epsilon}{\ln 1-x})$ approaches 0 implying that the second group can be ignored for sufficiently large w. Thus, as long as ϵ' is small enough such that $Q_i(w, l, d)$ is sufficiently small (equivalent to requiring w to be sufficiently large), then the odds of E_1 will be negligible and the gap between T_n and T_c can be arbitrarily small.

∎

Proof of Eq. (7): If $l = 1$, all mini-servers form into w clusters and each cluster with $\frac{m}{w}$ mini-servers replicating the same segment. Because the uploading rates are assumed to be i.i.d. variables, C_j the capacity to distribute segment j approximately follows the normal distribution with mean $\frac{m}{w}u$ and variance $\frac{m}{w}\sigma^2$. Then, the average file downloading time is

$$E[T_n] = \int_0^{+\infty} Pr(T_n > t) dt.$$

Let $C_{min} = min\{C_1, \ldots, C_w\}$, then $T_n = \frac{nF}{wC_{min}}$ and we have

$$Pr(T_n > t) = Pr\left(C_{min} < \frac{nF}{wt}\right).$$

The value of $Pr\left(C_{min} < \frac{nF}{wt}\right)$ can be derived as follow. Let G denote the CDF of any variable $C_1, \ldots, C_w$ since they have the identical distribution, then

$$Pr\left(C_{min} < \frac{nF}{wt}\right) = \sum_{j=1}^{w} Pr\left(C_{min} < \frac{nF}{wt} | C_{min} = C_j\right),$$

$$= w \int_0^{\frac{nF}{wt}} g(y) (1 - G(y))^{w-1} dy.$$

Here, G and g are determined as follows. Given that C_j is the sum of $\frac{m}{w}$ variables with the identical distribution, $\frac{C_j - \frac{m}{w}u}{\sqrt{\frac{m}{w}}\sigma}$ approximately follows standard normal distribution $\mathcal{N}(0, 1)$. Thus, $g(y) = \frac{\sqrt{w}}{\sqrt{m}\sigma} \phi\left(\frac{wy - mu}{\sqrt{mw}\sigma}\right)$ and $G(y) = \Phi\left(\frac{wy - mu}{\sqrt{mw}\sigma}\right)$, where ϕ and Φ are the PDF and CDF of standard normal distribution, and u and σ are the mean and standard deviation of U_j. Thus,

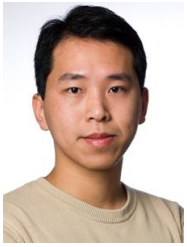
$$E[T_n] = \int_0^{+\infty} \int_0^{\frac{nF}{wt}} w g(y) (1 - G(y))^{w-1} dy dt,$$

$$= \int_0^{+\infty} \frac{nF}{y} g(y) (1 - G(y))^{w-1} dy.$$

Here $w g(y) (1 - G(y))^{w-1}$ is just the probability that $C_{min} < \frac{nF}{wt}$. ∎

Yipeng Zhou is a research fellow with the Institute for Telecommunications Research with University of South Australia. He was an assistant professor with the College of Computer Science and Software Engineering, Shenzhen University from 2013.9 to 2016.8. He received his B.S. degree from department of Computer Science of the University of Science and Technology of China (USTC), M.Phil. and Ph.D. degrees from the department of Information Engineering of the Chinese University of Hong Kong (CUHK) respectively. From 2012 to 2013, he was a postdoctoral research fellow with the Institute of Network Coding of CUHK. His main research interests lie in modeling and analysis of large scaled networking systems, analysis of user behaviors of online video systems and crowdsourcing-based content distribution.

Terence Chan received his B.Sc (Math), Master's and Ph.D. degrees in Information Engineering in 1996, 1998 and 2000 respectively, all from The Chinese University of Hong Kong. In 2001, he was a visiting assistant professor in the Department of Information Engineering at the same university. From February 2002 to June 2004, he was a Post-doctoral Fellow at the Department of Electrical and Computer Engineering at the University of Toronto. He was an assistant professor in University of Regina from 2004-2006. He is currently an Associate Professor in Institute for Telecommunications Research at University of South Australia.



Siu-Wai Ho received the B.Eng., M.Phil., and Ph.D. degrees in information engineering from The Chinese University of Hong Kong, in 2000, 2003, and 2006, respectively. During 2006âĂŞ2008, he was a Postdoctoral Research Fellow in the Department of Electrical Engineering, Princeton University, USA. Since 2009, he has been with the Institute for Telecommunications Research, University of South Australia, Australia, where he is currently a Senior Research Fellow. Dr. Ho received the Croucher Foundation Fellowship for 2006âĂŞ2008, the 2008 Young Scientist Award from the Hong Kong Institution of Science, and the Australian Research Council Australian Postdoctoral Fellowship for 2010âĂŞ2013. His paper received the best student paper award in the 2016 Australian communication theory workshop. His projects received the 2016 national iAward - consumer category from the Australian Information Industry Association and an honorary mention in the 2015 ComSoc Student Competition "Communications Technology Changing the World" organised by IEEE Communications Society.

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.