Scalable Video Streaming with Helper Nodes using Random Linear Network Coding

Pouya Ostovari*, Jie Wu*, Abdallah Khreishah†, and Ness B. Shroff‡
*Department of Computer & Information Sciences, Temple University, Philadelphia, PA 19122
†Department of Electrical & Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102
‡Department of Electrical & Computer Engineering, Ohio State University, Columbus, OH 43210

***Abstract*—Video streaming generates a substantial fraction of the traffic on the Internet. The demands of video streaming also increase the workload on the video server, which in turn leads to substantial slowdowns. In order to resolve the slowdown problem, and to provide a scalable and robust infrastructure to support on-demand streaming, helper-assisted video-on-demand (VoD) systems have been introduced. In this architecture, helper nodes, which are micro-servers with limited storage and bandwidth resources, download and store the user-requested videos from a central server to decrease the load on the central server. Multi-layer videos, in which a video is divided into different layers, can also be used to improve the scalability of the system. In this paper, we study the problem of utilizing the helper nodes to minimize the pressure on the central servers. We formulate the problem as a linear programming using joint inter- and intra-layer network coding. Our solution can also be implemented in a distributed manner. We show how our method can be extended to the case of wireless live streaming, in which a set of videos is broadcast. Moreover, we extend the proposed method to the case of unreliable connections. We carefully study the convergence and the gain of our distributed approach.**

***Index Terms*—Video-on-demand (VoD), streaming, multi-layer video, intra-layer coding, inter-layer coding.**

I. Introduction

Recent studies have shown that multimedia streaming produces a significant portion of the traffic on the Internet. For example, 20-30% of the web traffic on the Internet is from YouTube and Netflix [1], [2]. Thousands of hours of video are uploaded on YouTube every day, and millions of hours of movies are available on Netflix, Hulu, and iTunes sites.

In order to provide a scalable and robust infrastructure that will support large and diverse on-demand streaming, the concept of helpers has been introduced, and the design of *helper-assisted video-on-demand* (VoD) systems has been explored [3]–[7]. Helpers are micro-servers with limited storage and bandwidth resources, which can download and store requested videos to be able to serve user requests on demand. The helpers work in conjunction with a central server, which provides users with video files that cannot be obtained from their neighboring helpers (Figure 1). It is clear that the central server will be able to serve more users, as long as we can provide more portions of the requested videos through the helpers.

In addition to the use of helpers, we can benefit from *multi-layer videos* [8]–[11] to provide a higher degree of scalable VoD systems. In multi-layer video, which is also called *multi-resolution codes* (MRC) or *scalable video coding* (SVC) [12], [13], videos are typically divided into a base layer and enhancement layers [14], [15]. The base layer (layer 1) is required to watch the video, but the enhancement layers augment the quality of the video streaming. Accessing more layers provides higher video quality, but the i-th enhancement layer is not useful unless the user has access to all of the enhancement layers with a smaller index. Figure 2(a) shows an original image, and Figures 2(b)-(d) show the constructed layers from this image. Layer 1 is the most important layer, which is required by all the users. Layers 2 and 3 cannot be used without all of the layers with smaller indices, as depicted in Figures 2(c) and (d). Figure 2(f) shows that adding layers 2 and 3 together without layer 1 is useless, as well. Adding layer 2 to layer 1 increases the quality of the image, as shown in Figure 2(e).

In order to optimally use the resources, we need a mechanism to distribute the packets of the videos on the helpers, since, due to storage limitations, the helpers might not be able

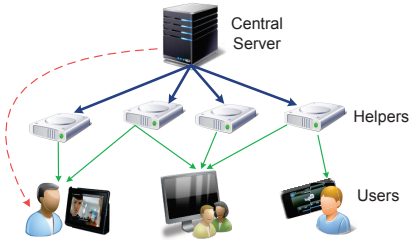


Fig. 1. The system architecture.

(a) Original (b) Layer 1 (c) Layer 2

(d) Layer 3 (e) Layers 1 & 2 (f) Layers 2 & 3

Fig. 2. Multi-layer video with 3 layers.

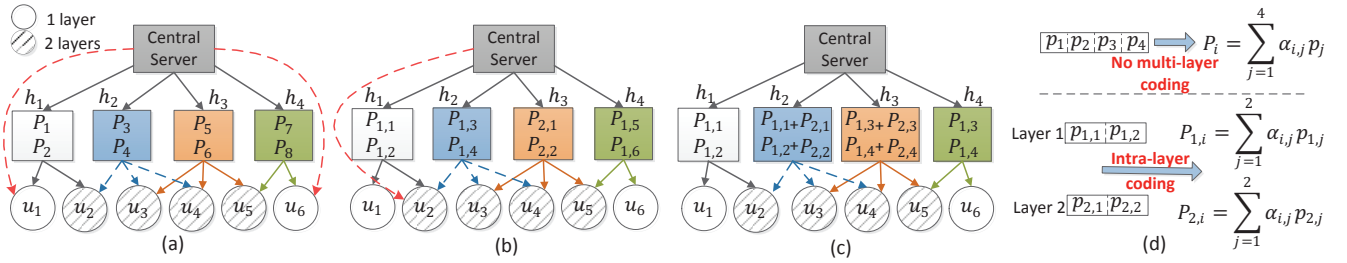


Fig. 3. The advantage of using NC, (a) No multi-layer NC, (b) Intra-layer NC, (c) Inter- & intra-layer NC, (d) Coding schemes.

to store a full copy of the video. Network coding (NC) [16]–[19] helps to simplify the content distribution problem, and solves it in an efficient way [20]. Consider packets $p_1, ..., p_n$. In *linear NC*, each coded packet is in the form of $\sum_{i=1}^{n} a_i \times p_i$, where a_i is a coefficient. In this scheme, if a user has access to any n linearly independent coded packets, it can use Gaussian elimination to decode the coded packets and retrieve the original packets. In [21], it is shown that when the coefficients are selected randomly, there is a very high probability that the packets will be linearly independent. As a result of this scheme, which is called *random linear NC*, the coded packets contribute the same amount of data to the users, which simplifies the distribution of the packets. Linear NC can be classified into *intra- or inter-layer NC*, depending on whether the coding is performed between the packets from the same layer or different layers, respectively.

Consider Figure 3, in which the users request a two-layer video, each of which consists of 2 packets; thus, the video contains 4 packets. The capacity of the helpers is equal to 2 packets. Assume that users u_1 and u_6 request layer 1, and that the other users need both layers. If we were to use the method in [4], the whole video would need to be downloaded for playing, as the method does not support multi-layer coding; thus, the video is considered to be 4 packets, p_1-p_4. Figure 3(a) shows an optimal video placement option based on the proposed method in [4], in which random linear coded packets of p_1-p_4 are stored on the helpers (the no multi-layer NC is depicted in Figure 3(d)). In this case, users u_2-u_5 have access to 4 coded packets over p_1-p_4; thus, they can decode the coded packets using just the helpers. However, users u_1 and u_6 need to download 2 more packets from the server to decode the coded packets.

Figure 3(b) shows an optimal placement using intra-layer NC. The coding structure is shown in Figure 3(d). In this case, only user u_2 needs to download 2 packets from the server, so the load on the server is less than that of in Figure 3(a). Inter-layer NC can be used in conjunction with intra-layer NC to increase the efficiency of the content placement on the helpers. In Figure 3(c), we benefit from inter-layer NC. Users u_2-u_5 have access to 4 linearly coded packets over layers l_1 and l_2, so the server does not need to upload any layer. Moreover, users u_1 and u_6 have access to 2 linearly coded packets over layer l_1, which is sufficient for decoding the first layer.

Motivated by the intuition drawn from the example, in this work, we answer the following questions: how should the packets of videos be distributed over helpers? How should the helpers allocate their bandwidth to the users to minimize the load on the central server? And, lastly, how should we design a coding scheme for content placement? While answering these questions, we make the following contributions:

- We study video streaming using helpers in the case of multi-layer multi-videos, and characterize the optimal solution using linear programming (LP).
- The problem of inter-layer NC is in general an NP-complete problem [9], [14]. However, in our problem, the optimal solution in the case of using triangular inter-layer NC can be calculated in polynomial time. We also present a distributed approach to optimally utilize the helpers, which adapts to the changes in the requested videos and the joining or departure of the nodes (helpers and users).
- We empirically show the cases under which combining inter- with intra-layer coding provide benefits (reduced server load) over intra-layer coding.
- In contrast with the work in [4], we extend our solutions to consider the reliability of the links.

The remainder of this paper is organized as follows: We review the related works in section II. In Section III, we introduce the settings. We formulate the problem for the case of wireless or wired VoD in Section IV, and we extend our proposed method to the case of networks with unreliable links in Section V. We study the wireless live streaming application in Section VI. In Section VII our distributed solution is proposed. We evaluate our methods through simulations in Section VIII. Section IX concludes the paper.

II. Related Work

Peer-to-peer (P2P) streaming has been studied in [22]–[24]. The authors in [22] study rate allocation problem in P2P VoD streaming. They propose a distributed rate allocation algorithm, which can reduce the unfriendly traffic to the Internet service providers (ISP), such as inter-ISP traffic, without much increase on the server load. In [23], distributed bandwidth allocation in live P2P streaming is studied. The challenges and the design issues of a large-scale P2P-VoD system are studied in [24]. The authors argue that less synchrony in the video contents shared by the users in VoD streaming makes the problem of reducing server load and maintaining streaming performance hard. In order to resolve this problem, each peer needs to contribute a small amount of storage. The paper proposes content replication, content discovery, and peer scheduling schemes.

The role of helpers in video streaming has been studied in several works. In [6], the authors study the live streaming of a single video in a helper-assisted P2P system. In their proposed method, each helpers downloads one coded packet of

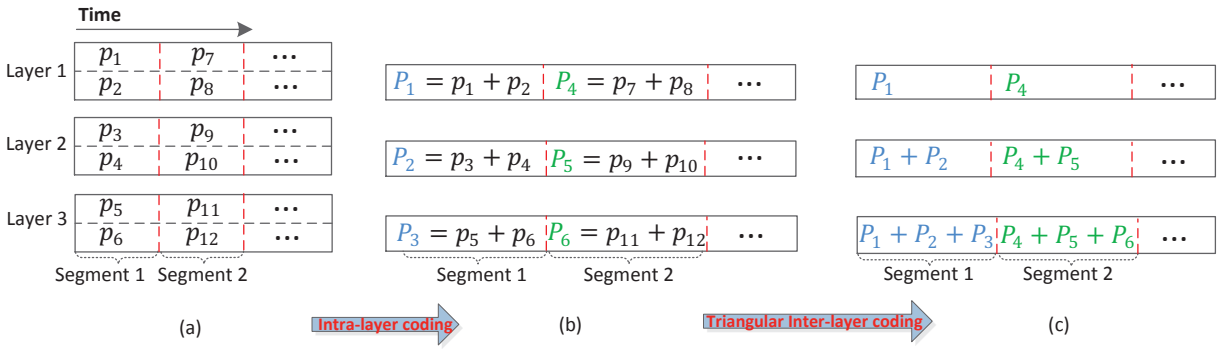


Fig. 4. (a) Segmentation of a multi-layer video with 3 layers. (b) Intra-layer NC. (c) Joint inter- and intra-layer coding.

the currently streamed segment. The simulation results show a significant increase in the streaming bitrate. The authors in [7] use helpers in a P2P VoD system to stream a single video. The authors propose a distributed bandwidth allocation algorithm for the helpers.

In [3], the role of coding in the design of a large-scale Video-on-Demand (VoD) system is studied. The authors show that NC can convert a combinatorial problem into a tractable problem. In [4], a P2P Video-on-Demand (VoD) system using helpers is proposed. The objective of the paper is to minimize the server load in the case of limited helpers' bandwidth and storage. The authors formulate the problem as an LP optimization, and propose a distributed algorithm to solve it. However, their distributed scheme oscillates among different solutions. As a result, it does not converge to the optimal solution. The other problem with the oscillation is that it can result in delay oscillation, which might cause playback lags. In contrast with the proposed distribution algorithm in [4], our distributed algorithm converges to the optimal solution very quickly. Moreover, in this work, we consider multi-layer VoD streaming and the unreliability of the links.

III. Setting

We consider a VoD system, where a video provider delivers a set of videos to a set of users. This video provider might consists of a set of video servers. In the rest of the paper we refer to this vide provide as the central server. A group of helpers, which are micro-servers with limited storage and bandwidth resources help the central server in providing the users with the videos. These helpers might be set up by the video providers or third-party companies in order to reduce the workload on the server. In general, the users can also participate in the video distribution by allocating a portion of their local storage to work as a helper. Without loss of generality, we consider the helpers and users to be separate nodes. We represent the set of helpers, users, and videos as H, U, and M, respectively. The users are stationary, and each helper covers a subset of the users (these sets does not need to be mutually exclusive). The coverage can be based on the geographic location or physical connection between the nodes. The k-th video m_k has a constant streaming rate r_k and size v_k. User u_i has a stationary request, denoted as q_i, which means u_i watches a single video at a time from the beginning to the end.

The helper h_j has storage and upload bandwidth capacities equal to S_j and B_j, respectively. If the helpers adjacent to user u_i can cumulatively provide the streaming rate of the requested video by the user, the whole video will be downloaded only from the helpers. Otherwise, the user will request the remaining portion of the video directly from the central server (Figure 1), which incurs some costs. In our problem, the cost is in terms of the load on the central server. Our objective in this work is to minimize the server's total upload rate to the users. In other words, we want to maximize the total number of videos that helpers provide to their adjacent users.

The users have diverse network conditions and use different types of devices with different processing and bandwidth resources to watch the videos. As a result, they might desire different levels of video quality. In order to provide the users with different levels of video qualities, each video m_k is divided into e_k layers. The l-th layer of video m_k has a streaming rate and size equal to r_{kl} and v_{kl}, respectively. Each user u_i can subscribe to his or her desired number of layers c_i. Requesting more layers results in a better watching quality. The user can make a decision regarding c_i based on the quality of its network connection, or any other network limitations it may have. The l-th layer of a video is not useful unless all of the layers with a smaller index are available. Let the j-th helper node's upload rate to its adjacent user u_i over the l-th layer of video m_k be x_{ji}^{kl}. We represent the set of adjacent helpers to the user u_i and adjacent users to the helper h_j as $N(u_j)$ and $N(h_j)$, respectively. We consider the links to be reliable. Later, in Section V, we extend our proposed methods to the case of unreliable connections. Table I summarizes the set of symbols used in this paper.

We develop a distributed algorithm to find the optimal solution for the stationary case, where the number of users and helpers are fixed. We show via simulation results that, even for dynamic networks, our algorithm appears to converge to the optimal solution.

IV. VoD with Multi-Layer Videos

In general, a helper might not be able to store a full copy of a video because of storage limitations. Moreover, a helper might provide more help to the central server by storing more partial videos, rather than by storing a small number of full videos [3]. The reason is that, by storing more partial videos, the helper can provide partial help to a greater number of users. Under this setting, in order to minimize the pressure on the central server, the following questions have to be addressed:

TABLE I
THE SET OF SYMBOLS USED IN THIS PAPER.

Notation	Definition
u_i, U	The i-th user, the set of users
h_j, H	The j-th helper, the set of helpers
m_k, M	The k-th video, the set of videos
B_j	The bandwidth limit of helper h_j
S_j	The capacity limit of helper h_j
r_{kl}/v_{kl}	The rate/size of layer l of video m_k
$N(u_i)/N(h_j)$	The set of adjacent helpers/users to u_i/h_j
x_{ji}^{kl}	Upload rate from h_j to u_i over layer l of video m_k
f_j^{kl}	The fraction of layer l of video m_k stored on helper h_j
e_k	The number of layers of video m_k
q_i	The requested video by user u_i
c_i	The number of layers requested by user u_i
x_j^k	Upload rate of helper h_j over video m_k (in live streaming)
d_{ji}^k	The download rate of user u_i from helper h_j over video m_k (in live streaming)
ϵ_{ji}	The reliability of the link between the helper h_j and user u_i
ϵ_i	The reliability of the link between server and user u_i

- *Content placement*: Which packets of which layers of each video should a helper store?
- *Bandwidth allocation*: Which packets, and to which adjacent users, should each helper serve its stored content?
- *Coding scheme*: How should we design the coding scheme for the helpers?

Intra-layer NC helps to simplify the content placement problem on the helpers. As stated in the introduction, intra-layer NC also increases the efficiency of the content placement on the helpers. We divide each layer of a video into segments of n packets, according to the playing time of the video frame that they belong to. Figure 4(a) shows a video with 3 layers. Each segment of a layer consists of 2 packets, and the playback of segment 1 is before segment 2. In our intra-layer NC scheme, each coded packet of a segment is a random linear combination of the whole packets in that segment. In Figure 4(b), the coefficients are not shown for simplicity. For instance, $p_1 + p_2$ means $a_1p_1 + a_2p_2$, where a_i is a random coefficient. Coding too many packets together increases the time and memory complexity of coding and decoding. That is why we partition the packets into segments and code the packets of each segment together. When using intra-layer NC, all of the coded packets from the helpers will contribute the same amount of information, and a user will be able to view the segment if it downloads any n linearly independent coded packets from the helpers that have the segment stored.

In order to enable a helper to serve any users watching video m, regardless of their playback time, we uniformly store the packets from each segment. Using this scheme, in order to store a fraction f of a layer of video m on helper h, we store $f \times n$ random linearly coded packets of each segment on the helper. Consider the video layer in Figure 5(a), in which each segment contains 4 packets. Assume that we want to store half of the video layer on a helper. We store 2 random linearly coded packets for each segment, as shown in Figure 5(b). Note that the 2 coded packets of each segment are different, since they have different random coefficients. For simplicity, we do not show the coefficients in the figure. Using this scheme, helper h can supply at the rate of $f \times r$ to the users that need video m, where r is the rate of the video. The use of intra-layer NC enables a flow-based model of the content, which changes our content placement and bandwidth allocation questions to finding:

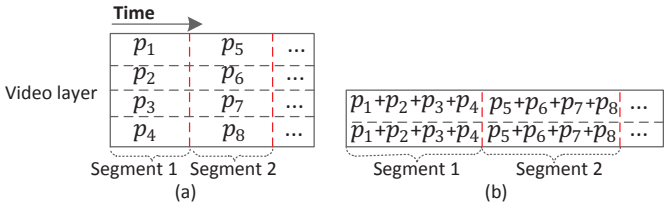


Fig. 5. (a) Segments of a video layer. (b) Storing a half of the video layer on a helper.

- The rate at which coded packets of a video layer should be stored on a helper.
- The rates at which coded packets of a video layer should be uploaded to a helper's adjacent users.
- The optimal coding scheme.

An alternative coding approach to random linear NC are rateless (fountain) codes [25], [26], which have less coding and decoding complexity. However, assuming that n packets are coded, $n(1+\epsilon)$ coded packets are required for decoding, where ϵ is a small overhead. It is shown that as n becomes larger, ϵ becomes smaller. In this paper we user random linear NC. However, it can be replaced with a rateless code.

A user might receive the packets of the current segment from the helpers and the server with different delays. As the packets of each segment are linearly coded with each other, the user is not able to decode the segment until it receives enough coded packets for the current segment. It means that, the user will be able to decode and watch the current segments when it received the last coded packet of the current segment from the helper or server that has the largest delay. In order to address this problem, which might result in the video lag problem (the video stopping as a result of delay in receiving the required packets), each user buffers the received coded packets and delays the playback so that the differences of the transmission delays and changes in the delays do not result in playback lags. The delay of the paths changes over time; thus; this buffering time should be large enough such that it does not result in lag problem. A large buffering time increases the waiting time to start the payback; thus, the buffering time should not be too large. Computing the buffering time of the users is beyond the scope of this work.

A. Intra-Layer Network Coding

In this section, we assume that the links are reliable. As a result, minimizing the server load is equivalent to maximizing the help provided by the helpers. This optimization can be modeled as the following LP optimization problem:

We refer to this problem as problem A. Objective function (1) is the summation of the helper nodes' upload rates to their adjacent users, over the subscribed layers of the requested videos. Function (1) is a linear function, so it is a concave

$$\max \sum_{i,k:u_i\in U, m_k=q_i} \sum_{j,l:h_j\in N(u_i), l\le c_i} x_{ji}^{kl} \quad (1)$$

$$s.t \quad x_{ji}^{kl} \le f_j^{kl} r_{kl}, \quad \forall j,i,l,k : u_i \in N(h_j), m_k = q_i, l \le e_k \quad (2)$$

$$\sum_{i,k:u_i\in N(h_j), m_k=q_i} \sum_{l\le c_i} x_{ji}^{kl} \le B_j, \quad \forall j : h_j \in H \quad (3)$$

$$\sum_{k:m_k\in M} \sum_{l:l\le e_k} f_j^{kl} v_{kl} \le S_j, \quad \forall j : h_j \in H \quad (4)$$

$$\sum_{j:h_j\in N(u_i)} x_{ji}^{kl} \le r_{kl}, \quad \forall i,l,k : u_i \in U, l \le c_i, m_k = q_i \quad (5)$$

$$0 \le f_j^{kl} \le 1, \quad \forall j,k,l : h_j \in H, m_k \in M, l \le e_k \quad (6)$$

Fig. 6. Problem A: LP optimization with intra-layer NC.

function (Note that (1) is not strictly concave). We use the set of Constraints (2) to limit each helper's upload rate at the available service rate of the videos (the rate of stored videos). This upload rate differs for different layers of a video; thus, for each layer of a video, we have a separate constraint. The set of constraints (3) and (4) are feasibility constraints on bandwidth and storage, respectively. In more detail, the total upload rate of a helper and the total stored data on it cannot exceed its bandwidth and capacity limit. Note that in VoD applications, even in the case that the adjacent users to a helper watch the same video, their playback times are different; so, the helper needs to allocate separate bandwidths for each adjacent user. If two adjacent users to the same helper are watching the same video within a small time difference (e.g, 30 seconds), buffering at the users could be used to transmit the same information to both. However, for long videos it does not happen frequently.

It is sufficient for user u to download layer l of its requested video m_k at a rate equal to the streaming rate of the layer, since more than that value will not be useful. The set of Constraints (5) limits the aggregated download rate of the requested layers of video m to the user u at the rate of the layers. The set of Constraints (6) are the feasibility constraints on the fraction of stored video layers on the helpers, which limits them to be in the range of 0 and 1.

Assuming that each user is connected to all of the helpers, the number of variables x and f are equal to $|U||H|e$ and $|H||M|e$, where e is the maximum number of video layers. Moreover, the number of Constraints (2)-(6) are equal to $|U||H|e$, $|H|$, $|H|$, $|U|e$, and $|H||M|e$, respectively. Therefore, the solution of the optimization can be calculated in polynomial time [27].

B. Joint Inter- and Intra-Layer Network Coding

In this section, we extend the formulation of problem A (Figure 6) to the case of joint inter- and intra-layer NC.

In the general form of random linear NC, each packet can be coded with any other packets. Thus, in the case of n packets, there are $2^n - 1$ random linear NC possibilities. Figure 7(b) shows the seven possible ways to code the three packets in Figure 7(a) using the general form of NC. For simplicity, the random coefficients are not shown in the figures. In contrast with the general form of coding, in *triangular NC* [28], each coded packet is a random linear combination of the first i packets, $\forall i : 1 \le i \le n$. In other words, the coded packets have a prefix form. Therefore, there are just n possibilities for coding n original packets. Figure 7(c) shows the three possible coded packets using the triangular coding scheme.

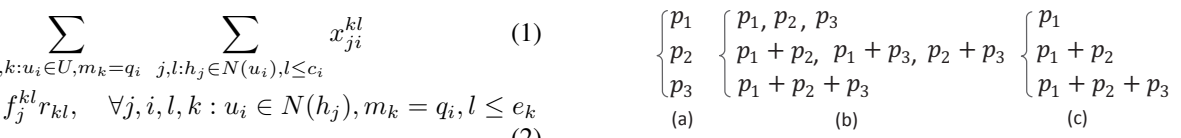


Fig. 7. p_1+p_2 means $a_1p_1+a_2p_2$, where a_i is a random coefficient. (a) The original packets. (b) The general form of random linear NC. (c) Triangular NC scheme.

As stated in the introduction, inter-layer NC helps to increase the provided help of the helpers. In order to benefit from joint inter- and intra-layer NC, we first perform intra-layer NC (Figure 4(b)). Then, we use the triangular NC scheme to code the intra-layer coded packets together. In our scheme, the coded packets of each segment of a video's l-th layer are a random linear combination of that segment in layers 1 to l. Figure 4(c) depicts the joint inter- and intra-layer coded packets, using the triangular scheme. In this figure, the packets of layer 1 are similar to those in the intra-layer approach, but the packets of layer 2 are a linear combination of layers 1 and 2. Also, the packets of layer 3 are a random linear combination of all the 3 layers. In the figure, the random coefficients are not shown for simplicity. For example, P_1+P_4 means $a_1P_1 + a_2P_4$.

We prefer using triangular NC over the general form for two reasons. First, it limits the coding space of the coding problem, such that the convex optimization problem can be solved in a polynomial time. Second, in our setting, we limit the number of received layers of each user to his request. Because of this limitation, the gain of the triangular NC is not less than the general form of NC. Before discussing the reason, we propose the following lemma:

Lemma 1: A set of general linear coded packets (non-triangular) can be mapped to a set of triangular coded packets such that the rank of the set is preserved.

Proof is provided in Appendix A. Under the proposed setting, we do not provide a user with more number of layers that he requested, which makes the gain of the triangular NC not less than that of the general form of inter-layer NC. Assume that the largest non-zero index in a general coded packet is d. Based on our setting, we should deliver c_i layers to user u_i. As a result, we do not transmit this coded packet to a user that requested fewer layers than d layers, and any changes in this coded packet does not have any impact on this user. On the other hand, following Lemma 1, mapping a set of general linear coded packets to a set of triangular coded packets does not change the rank of the set. Therefore, the mapping does not have a negative impact on the users that requested at least d layers.

Assume that user u_i has subscribed to c_i layers, each of which contains n packets. We represent the received coded packets of the l-th coded layer as Z_l. In [28], it is shown that under the triangular coding scheme, a user can decode all of the c_i layers if $\sum_{j=c_i-l+1}^{c_i} |Z_j| \geq ln,\ \forall l \in [1, c_i]$. This means that the total number of received coded packets should be at least equal to $c_i n$. Also, the total number of received coded packets from layers 2 to c_i needs to be equal to or more than $(c_i - 1)n$. In general, the number of received coded packets from layers l to c_i should not be less than $(c_i - l + 1)n$, which gives us an insight into the following lemma:

Lemma 2: Assume that the l-th layer contains n_l packets. Providing more than $\sum_{l=1}^{l'} n_l$ coded packets from the first l' coded layers is not useful to user u.

For the proof refer to Appendix B. From Lemma 2, for the case of joint inter- and intra-layer NC, we can modify the formulation of problem A in Figure 6, as Follows. The objective function is the same as that of in problem A. Also, much like the problem A, we have the set of Constraints (2), (3), and (4). However, the set of Constraints (5) should be modified as:

$$\sum_{l=1}^{l'} \sum_{j:h_j \in N(u_i)} x_{ji}^{kl} \leq \sum_{l=1}^{l'} r_{kl}, \quad \forall i, l', k : 1 \leq l' \leq c_i, u_i \in U, m_k = q_i \tag{7}$$

which implies that the total upload rates of the first l' layers of the user-requested video should not be more than the total streaming rate of those layers. This set of constraints ensures that the helpers will not provide users with coded packets that are not useful for decoding. Moreover, we do not have the set of Constraints (6) anymore. The reason is that the coded video layers are joint inter- and intra layer. As a result, the variable f can be greater than 1. For example, consider a 2-layer video with n packets per layer. In the case of inter- and intra-layer coding, a user that does not receive any coded packet from layer 1 and receives $2n$ coded packets over layers 2 is able to decode both of the layers. Consequently, a helper can store the inter-layer coded packets of layer 2 at a rate equal to $f = 2$.

V. VoD with Unreliable Connections

We extend our methods for unreliable connections in two cases. In the first case we assume that the connections between the server and the users are reliable, and that losses happen just on the links between the helpers and the users. In the second case, the links from the server and helpers to the users might be lossy.

A. Reliable Server Links

As we assume that the links from the server to the users are reliable, maximizing the provided data to the users by the helpers minimizes the load on the server. The main difference between the case of unreliable helper links and the proposed methods in Section IV is that, here, the provided data (delivered successfully) from helper h_j to its neighboring user u_i over layer k is equal to $\epsilon_{ji} x_{ji}^{kl}$, where ϵ_{ji} is the reliability of the link between these two nodes. Based on this discussion, in the following sections, we modify the proposed linear programming equations in Section IV.

1) VoD with intra-layer Coding: In the case of using intra-layer coding and the existence of lossy helper nodes' links, the optimal bandwidth and storage can be found using the following LP:

$$\max \sum_{i,k:u_i \in U, m_k = q_i} \ \sum_{j,l:h_j \in N(u_i), l \leq c_i} \epsilon_{ji} x_{ji}^{kl} \tag{8}$$

$$s.t \quad x_{ji}^{kl} \leq \frac{f_j^{kl}}{\epsilon_{ji}} r_{kl}, \forall j, i, k, l : u_i \in N(h_j), m_k = q_i, l \leq e_k \tag{9}$$

$$\sum_{i,k:u_i \in N(h_j), m_k = q_i} \ \sum_{l \leq c_i} x_{ji}^{kl} \leq B_j, \quad \forall j : h_j \in H \tag{10}$$

$$\sum_{k:m_k \in M} \sum_{l:l \leq e_k} f_j^{kl} v_{kl} \leq S_j, \quad \forall j : h_j \in H \tag{11}$$

$$\sum_{j:h_j \in N(u_i)} \epsilon_{ji} x_{ji}^{kl} \leq r_{kl}, \forall i, l, k : u_i \in U, l \leq c_i, m_k = q_i \tag{12}$$

$$0 \leq f_j^{kl} \leq 1, \quad \forall j, k, l : h_j \in H, m_k \in M, l \leq e_k \tag{13}$$

The objective function (8) is a summation of the delivered data from the helpers to their adjacent users. In contrast with the case of reliable links, here x_{ji}^{kl} is not equal to the received rate of the user u_i. As a result, the helpers can perform redundant transmission, since some of the transmissions will be lost. That is why, in the set of Constraints (9), we divide f_j^{kl} by ϵ_{ji}. The set of Constraints (10) and (11) ensure that the data uploaded by a helper and the stored data on the helper do not exceed its bandwidth and storage constraints. We use the set of Constraints (12) to limit the receiving rate of each layer of a video by a user, which is equal to $\epsilon_{ji} x_{ji}^{kl}$, to the rate of that layer. We refer to this LP as reliable intra-layer coding.

2) VoD with Joint Inter and Intra-layer Coding: Similar to the case of using intra-layer NC, we can modify the proposed joint inter- and intra-layer LP to the case of unreliable helper links. Solving the following LP results in the optimal bandwidth and storage allocation when the links between helpers and the users are unreliable:

$$\max \sum_{i,k:u_i \in U, m_k = q_i} \ \sum_{j,l:h_j \in N(u_i), l \leq c_i} \epsilon_{ji} x_{ji}^{kl}$$

$$s.t \quad x_{ji}^{kl} \leq \frac{f_j^{kl}}{\epsilon_{ji}} r_{kl}, \ \forall j, i, k, l : u_i \in N(h_j), m_k = q_i, l \leq e_k$$

$$\sum_{i,k:u_i \in N(h_j), m_k = q_i} \ \sum_{l \leq c_i} x_{ji}^{kl} \leq B_j, \quad \forall j : h_j \in H$$

$$\sum_{k:m_k \in M} \sum_{l:l \leq e_k} f_j^{kl} v_{kl} \leq S_j, \quad \forall j : h_j \in H$$

$$\sum_{l=1}^{l'} \sum_{j:h_j \in N(u_i)} \epsilon_{ji} x_{ji}^{kl} \leq \sum_{l=1}^{l'} r_{kl}, \forall i, l', k : 1 \leq l' \leq c_i, m_k = q_i$$

B. Lossy Server Links

In the previous sections we maximized the provided data from the helpers to the users, since that is equivalent to

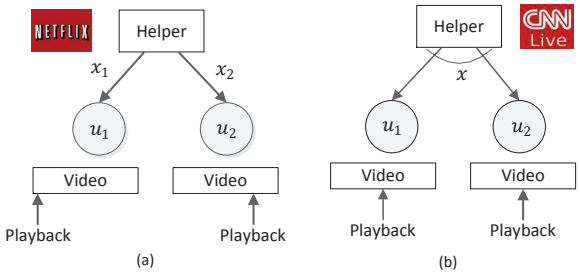


Fig. 8. VoD vs. Live streaming.

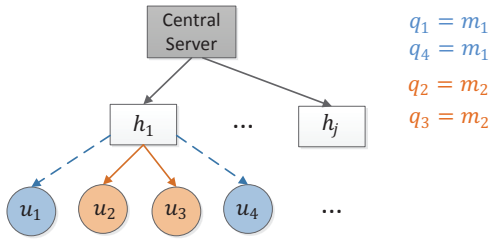


Fig. 9. Live streaming.

minimizing the load on the server. It is obvious that when the links between the server and users are lossy, maximizing the provided data from the helpers to the users might not minimize the load on the server. The reason is that, the links between the server and the users that could not download a 100% portion of their requested video from the helpers might be worth much more than the users that receive their requests in full from the helpers. As a result, we need to change the objective functions of the proposed linear programming from maximizing the provided help from the helpers to minimizing the server load az follows:

$$\min \sum_{i,k,l:u_i\in U, m_k=q_i, l\leq c_i} \left[r_{kl} - \sum_{j:h_j\in N(u_i)} \epsilon_{ji} x_{ji}^{kl}\right]/\epsilon_i \quad (14)$$

We represent the delivery rate of the link between the user u_i and the central server as ϵ_i. The rate of the layer k is equal to r_{kl}, and the inner summation in Equation (14) is the total portion of the layer k provided to the user u_i through its neighboring helpers. Therefore, the summation over the users in Equation (14) results in the total load on the central server.

VI. Wireless Live Streaming Applications

In this section, we show how the proposed solution for VoD can be extended for wireless live streaming (LS) applications. By 'LS' we are referring to applications where some videos are broadcast to the users, such as TV station channels or surveillance systems. In VoD, the users can play the videos asynchronously as depicted in Figure 8(a). However, in LS, the playback times of the users that watch the same video are synchronous (Figure 8(b)). Therefore, the main difference between LS and VoD is that, in LS, the helpers do not need to allocate separate bandwidths to their adjacent users that watch the same video, as shown in Figure 8(b).

In the case of VoD, the summation of the allocated bandwidth from each helper to its adjacent users should be less than or equal to its bandwidth. However, in LS, the summation of the allocated bandwidth from each helper for all of the videos should be less than or equal to its bandwidth. The reason for this is that more than one neighboring user might request the same video, and all of the users use the same broadcast packets. Consider Figure 9, where users u_1 and u_4 requested the same video, m_1. Also, the users u_2 and u_3 requested video m_2. In this case, helper h_1 shares its bandwidth between videos v_1 and v_2, without assigning a separate bandwidth for each users. In order to formulate the case of LS, we represent the allocated bandwidth for the video m_k over helper h_j as x_j^k. The summation of these variables for each helper should be less than or equal to the helper's bandwidth. Also, the download rate of user u_i over video m_k from the helper h_j, which is represented as d_{ji}^k, should be less than or equal to x_j^k. The problem of LS in the case of single-layer videos can be formulated as follows:

$$\max \sum_{i,k:u_i\in U, m_k=q_i} \sum_{j:h_j\in N(u_i)} d_{ji}^k \quad (15)$$

$$s.t \quad x_j^k \leq f_j^k r_k, \quad \forall j,k : m_k \in M \quad (16)$$

$$\sum_{k:m_k\in M} x_j^k \leq B_j, \quad \forall j : h_j \in H \quad (17)$$

$$\sum_{k:m_k\in M} f_j^k v_k \leq S_j, \quad \forall j : h_j \in H \quad (18)$$

$$d_{ji}^k \leq x_j^k, \quad \forall i,j,k : h_j \in N(u_i), m_k = q_i \quad (19)$$

$$\sum_{k:m_k=q_i} \sum_{j:h_j\in N(u_i)} d_{ji}^k \leq r_k, \quad \forall i : u_i \in U$$

$$0 \leq f_j^k \leq 1, \quad \forall j,k,l : h_j \in H, m_k \in M, l \leq e_k$$

Objective function (15) is the summation of the download rates of users. The set of Constraints (16) ensures that the upload rate of a video by a helper cannot exceed the available service rate of the video. Constraints (17), (18), and (6) are feasibility constraints on bandwidth and storage. We limit the download rate of a user from a helper to the upload rate of its requested movie, using the set of Constraints (19). We refer to our method as wireless live streaming (WLS). For simplicity we formulated the problem for the case of single-layer videos. The formulation can be easily extended for multi-layer videos.

VII. Distributed Solution

In this section, we solve the proposed optimization problem A (Figure 6) for the case of multi-layer VoD streaming, using intra-layer NC in a distributed way. The same approach can be used to find a distributed solution for the other settings. The idea is to solve the Lagrange dual of the problem using the gradient method [29]. In this way, the helpers start from empty storage, and gradually update their storage and bandwidth allocation, based on the exchanged Lagrange variables between them and their adjacent users.

The objective function (1) is not a strictly concave function, due to the presence of a linear summation. Consequently, a direct application of standard gradient iterative method might lead to multiple solutions. In this case, the output of an iterative method may oscillate between multiple feasible solutions. In order to overcome the problem due to the lack of

strict concavity, we can apply the Proximal method described in [30], page 233. The idea behind the Proximal method is to add quadratic terms to the objective function and make it strictly concave. A detailed description of the Proximal method is in [30], [31]. To apply the Proximal method, we introduce auxiliary variables y_{ji}^{kl}. By using the Proximal method, the optimization becomes:

$$\max \sum_{i,k:u_i \in U, m_k = q_i} \; \sum_{j,l:h_j \in N(u_i), l \leq c_i} \left(x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{kl})^2\right) \tag{20}$$

subject to the set of Constraints (2), (3), (4), (5), and (6).

The optimal solution of (20) is also the solution of (1). Let $\vec{x^*}$ and $\vec{f^*}$ be the optimal solution of (1) then, $\vec{x} = \vec{x^*}$, $\vec{f} = \vec{f^*}$, and $\vec{y} = \vec{x}$ is the maximizer of (20). The standard proximal method iteratively works as follows:

1) Fix $\vec{y}(t)$ and maximize (20) with respect to variables $\vec{x}(t)$ and $\vec{f}(t)$.
2) Set $\vec{y}(t+1) = \vec{x}(t)$, increment t, and go back to step 1.

Since the Slater condition holds (see reference [32]), there is no duality gap between the primal and the dual problems. Therefore, we can use the dual approach to solve the problem. Let λ_1^{jil}, λ_2^j, λ_3^j, and λ_4^{il} be the Lagrange variables for Constraints (2), (3), (4), and (5), respectively. Here, i, j, and l are corespondent to the indices in the set of Constraints (2) to (5). The Lagrange function of (20) is:

$$\begin{aligned}
&L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda}) = \\
&\sum_{i,k:u_i \in U, m_k = q_i} \; \sum_{j,l:h_j \in N(u_i), l \leq c_i} \left(x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{hl})^2\right) \\
&- \sum_{j,i:h_j \in H, u_i \in N(h_j)} \; \sum_{k,l:m_k = q_i, l \leq c_i} \lambda_1^{jil}(x_{ji}^{kl} - f_j^{kl} r_{kl}) \\
&- \sum_{j:h_j \in H} \lambda_2^j \Big(\sum_{i,k,l:u_i \in N(h_j), m_k = q_i, l \leq c_i} x_{ji}^{kl} - B_j \Big) \\
&- \sum_{j:h_j \in H} \lambda_3^j \Big(\sum_{k,l:m_k \in M, l \leq e_k} f_j^{kl} v_{kl} - S_j \Big) \\
&- \sum_{i,k,l:u_i \in U, m_k = q_i, l \leq c_i} \lambda_4^{il} \Big(\sum_{j:h_j \in N(u_i)} x_{ji}^{kl} - r_{kl} \Big)
\end{aligned}$$

By rearranging the terms, we have:

$$\begin{aligned}
&L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda}) = \sum_{i,k:u_i \in U, m_k = q_i} \; \sum_{j,l:h_j \in N(u_i), l \leq c_i} \big[(1 - \lambda_1^{jil} - \lambda_2^j \\
&- \lambda_4^{il}) x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{kl})^2\big] \\
&+ \sum_{j,i:h_j \in H, u_i \in N(h_j)} \; \sum_{k,l:m_k = q_i, l \leq c_i} \lambda_1^{jil} f_j^{kl} r_{kl} \\
&- \sum_{j:h_j \in H} \; \sum_{k,l:m_k \in M, l < e_k} \lambda_3^j f_j^{kl} v_{kl}
\end{aligned}$$

By a simple change of variables, the Lagrange function is separable in $\vec{x}$ and $\vec{f}$, and we can rewrite it as:

$$\begin{aligned}
&L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda}) = \\
&\sum_{\substack{i,k:u_i \in U \\ m_k = q_i}} \; \sum_{\substack{j,l:h_j \in N(u_i) \\ l \leq c_i}} \big[(1 - \lambda_1^{jil} - \lambda_2^j - \lambda_4^{il}) x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{kl})^2\big] \\
&+ \sum_{\substack{j,k:h_j \in H \\ m_k \in M}} \Big(\sum_{\substack{i,l:u_i \in N(h_j) \\ l \leq c_i}} \lambda_1^{jil} r_{kl} - \sum_{l:l < e_k} \lambda_3^j v_{kl} \Big) f_j^{kl}
\end{aligned} \tag{21}$$

The objective function of the dual problem is:

$$D(\vec{y}, \vec{\lambda}) = \max_{\substack{\vec{x} \geq 0 \\ \vec{y} \geq 0}} L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda})$$

The dual problem itself is $\min_{\lambda \geq 0} D(\vec{y}, \vec{\lambda})$. The dual optimization problem can be solved using the gradient method [29]. The updates of the Lagrange variables are listed as follows:

$$\lambda_1^{jil}(t+1) = \Big[\lambda_1^{jil}(t) + \alpha(x_{ji}^{kl}(t) - f_j^{kl}(t) r_{kl})\Big]^+,$$
$$\forall j, i, k, l : h_j \in H, u_i \in N(h_j), m_k = q_i, l \leq e_k$$

$$\lambda_2^j(t+1) = \Big[\lambda_2^j(t) + \alpha \sum_{i,k:u_i \in N(h_j), m_k = q_i} \; \sum_{l \leq c_i} x_{ji}^{kl}(t) - B_j\Big]^+,$$
$$\forall j : h_j \in H$$

$$\lambda_3^j(t+1) = \Big[\lambda_3^j(t) + \alpha \sum_{k:m_k \in M} \; \sum_{l:l \leq e_k} f_j^{kl}(t) v_{kl} - S_j\Big]^+,$$
$$\forall j : h_j \in H$$

$$\lambda_4^{il}(t+1) = \Big[\lambda_4^{il}(t) + \alpha \sum_{j:h_j \in N(u_i)} x_{ji}^{kl}(t) - r_{kl}\Big]^+,$$
$$\forall i, k, l : u_i \in U, m_k = q_i, l \leq c_i$$

where $[.]^+$ denotes the projection on $[0, \infty)$. Also, by setting the first derivative of (21) with respect to $\vec{x}$ being equal to zero, the optimal $\vec{x}$ can be calculated as follows:

$$x_{ji}^{kl}(t+1) = \frac{1 - \lambda_1^{jil}(t) - \lambda_2^j(t) - \lambda_4^{il}(t)}{2} + y_{ji}^{kl}(t)$$
$$\forall j, i, k, l : h_j \in H, u_i \in N(h_j), m_k = q_i, l \leq e_k$$

Algorithm 1 Calculation of $\vec{f}$ (for helper h_j)

```
rem = S_j, calculate γ_j^{kl}   ∀k, l : m_k ∈ M, l ≤ e_k
for each f_j^{kl} in descending order of γ_j^{kl} do
    if γ_j^{kl} > 0 and rem > 0 then
        if rem > v_{kl} then
            set f_j^{kl} = 1, rem = rem − v_{kl}
        else set f_j^{kl} = rem / v_{kl}, rem = 0
    else f_j^{kl} = 0, rem = 0
```

Algorithm 1 illustrates the computation of $\vec{f}$. Here, $\gamma_j^{kl} = \sum_{\substack{j,k:h_j \in H \\ m_k \in M}} (\sum_{\substack{i,l:u_i \in N(h_j) \\ l \leq c_i}} \lambda_1^{jil} r_{kl} - \sum_{l:l<e_k} \lambda_3^j v_{kl})$ is the multiplier of f_j^{kl} in Equation (21), and rem is the free space of helper h_j. The idea here is that, in order to maximize the second line in Equation (21), we should give a greater value to

Algorithm 2 Users' Protocol (for user u_i)

1: **Initialization**
2: Send the request and the number of desired layers to the adjacent helpers. Set $\lambda_4^{il}(1,0)=0$
3: **Iteration Phase** at the τ-th iteration
4: **for** $t=0,...,T-1$ perform the following step sequentially
5: send $\lambda_4^{il}(\tau,t+1) = [\lambda_4^{il}(\tau,t) + \alpha(\sum_{j:h_j\in N(u_i)} x_{ji}^{kl}(\tau,t) - r_{kl})]^+ \quad \forall l : l \le c_i$ to all adjacent helpers.
6: $\vec{\lambda_4}(\tau+1,0)=\vec{\lambda_4}(\tau,T)$

Algorithm 3 Helpers' Protocol (for helper h_j)

1: **Initialization**
2: Set $x_{ji}^{kl}(1,0)=0$, $f_i^{kl}(1,0)=0$, $\lambda_1^{jil}(1,0)=0$, $\lambda_2^{j}(1,0)=0$, $\lambda_3^{j}(1,0)=0$, $y_{ji}^{kl}(1,0)=0$
3: **Iteration Phase** at the τ-th iteration
4: **for** $t=0,...,T-1$ perform the following steps sequentially
5: $\lambda_1^{jil}(\tau,t+1) = [\lambda_1^{jil}(\tau,t)+\alpha(x_{ji}^{kl}(\tau,t)-f_j^{kl}(\tau,t)r_{kl})]^+$
6: $\lambda_2^{j}(\tau,t+1) = [\lambda_2^{j}(\tau,t) + \alpha\sum_{i:u_i\in N(h_j)}\sum_{l\le c_i} x_{ji}^{kl}(\tau,t) - B_j]^+$
7: $\lambda_3^{j}(\tau,t+1) = [\lambda_3^{jl}(\tau,t) + \alpha(\sum_{k:m_k\in M}\sum_{l:l\le e_k} f_j^{kl}(\tau,t)v_{kl} - S_j)]^+$
8: $x_{ji}^{kl}(\tau,t+1) = \frac{1-\lambda_1^{jil}(\tau,t)-\lambda_2^{j}(\tau,t)-\lambda_4^{il}(\tau,t)}{2} + y_{ji}^{kl}(\tau,t)$
9: run algorithm 1 to calculate $\vec{f}(\tau,t+1)$
10: $\vec{y}(\tau+1,0)=\vec{x}(\tau,T)$, $\vec{x}(\tau+1,0)=\vec{x}(\tau,T)$, $\vec{\lambda_1}(\tau+1,0)=\vec{\lambda_1}(\tau,T)$, $\vec{\lambda_2}(\tau+1,0)=\vec{\lambda_2}(\tau,T)$, $\vec{\lambda_3}(\tau+1,0)=\vec{\lambda_3}(\tau,T)$

the fraction of the videos with a greater γ value. On the other hand, the fraction of videos with a negative γ value should be equal to zero. Therefore, for each helper, we sort the γ_j^{kl} in descending order of their values, and we start to fill the helpers with videos that have a greater γ. Let us represent the set of videos requested by the neighboring users of helper h_j as M_j. For each layer of a video we have a f_j^{kl} variable; thus, the number of executions of the For loop in Algorithm 1 is equal to the total number of layers of the videos in M_j. Therefore, the time complexity of Algorithm 1 is order of $O(\sum_{k\in M_j} e_k)$. If we represent the maximum number of layers as e, the complexity will be is order of $O(|M_j|e)$.

We can define two iterative levels for the distributed algorithm [31]. In the inner loop, we fix the auxiliary variables $\vec{y}$ and update $\vec{x}$, $\vec{f}$, and $\vec{\lambda}$, for T times. We run the outer loop τ times, in which we set $\vec{y}(\tau+1,0)=\vec{x}(\tau,T)$. The users' and helpers' policies are shown in Algorithms 2 and 3, respectively. The users can receive x_{ij}^{km} from the helpers explicitly. However, they can compute it based on the actual receiving data rate from their neighboring helpers. The convergence of our algorithm can be proven using a technique similar to [33]. We omit the proof for brevity, and in our simulations, we empirically verify the convergence.

The For loop in Algorithms 2 runs for T iterations. Also, user u_i needs to calculate c_i different λ_4^{il} variables. For each of these Lambda variables, the the summation in line 5 has a complexity of $O(|N(u_i)|)$. Therefore, the complexity of Algorithms 2 is $O(|N(u_i)|c_iT)$. The For loop in Algorithms 3 runs for T iterations. Line 5 needs to be calculated for $|N(h_j)|e$, where e is the maximum number of video layers. The complexity of line 6, 7, and 8 is $O(e|M_j|)$. Also, in line 9, the helper node runs Algorithm 1. As a result, the complexity of Algorithms 3 is in order of $O(Te(|M_j|+|N(h_j)|))$.

Consider the topology in Figure 10, in which users u_1 and u_2 requested movies m_1 and m_2, respectively. In order to simplify the example, we assume that the movies contain one layer. As a result, we remove the index l from all of the notations and equations. Also, without loss of generality, we do not show variable τ in the equations, since we discuss the process for one round of updates. At the beginning, the users u_1 and u_2 set up their Lagrange variables λ_4^1 and λ_4^2 to zero, respectively. Moreover, helpers h_1 and h_2 initialize their correspondent variables x_{ji}^k, f_j^k, $\lambda_1^{j,i}$, λ_2^j, and λ_3^j to zero, or any other default value (Figure 10(a)). The variables y_{ji}^k are initialized to the values of their correspondent variables x_{ji}^k by the helpers.

Following the initializing phase, the users update their Lagrange variables λ_4^1 and λ_4^2 based on the received bandwidths x_{ji}^k from the helpers. User u_1 calculates $\lambda_4^1(t+1) = [\lambda_4^1(t) + \alpha(x_{1,1}^1(t) + x_{2,1}^1(t) - r_1)]^+$, and transmits it to the helpers h_1 and h_2. Similarly, as shown in Figure 10(b), user u_2 calculates $\lambda_4^2(t+1) = [\lambda_4^2(t) + \alpha(x_{1,2}^1(t) + x_{2,2}^1(t) - r_2)]^+$ and sends it to the helpers. In the next round, the helpers use the received Lagrange variables to update their Lagrange variables, and modify the bandwidth and storage assignment.

In Figure 10(c) we show the updates for helper h_1. The updates for helper h_2 can be done in a similar way. The helper h_1 first updates $\lambda_1^{1,1}$ and $\lambda_1^{1,2}$ using equations $\lambda_1^{1,1}(t+1) = [\lambda_1^{1,1}(t) + \alpha(x_{1,1}^1(t) - f_1^1(t)r_1)]^+$ and $\lambda_1^{1,2}(t+1) = [\lambda_1^{1,2}(t) + \alpha(x_{1,2}^2(t) - f_1^2(t)r_2)]^+$, respectively. These Lagrange variables correspond to the set of Constraints (2). Moreover, helper h_1 calculates λ_2^1 and λ_3^1 as shown in Figure 10(c). Then, the helper runs Algorithm 1 to calculate f_1^1 and f_1^2. Finally, helper h_1 assigns its bandwidth to the users u_1 and u_2, using equations $x_{1,1}^1(t+1) = [1-\lambda_1^{1,1}(t)-\lambda_2^1(t)-\lambda_4^1(t)]/2 + y_{1,1}^1(t)$ and $x_{1,2}^1(t+1) = [1-\lambda_1^{1,2}(t)-\lambda_2^1(t)-\lambda_4^1(t)]/2 + y_{1,2}^1(t)$, respectively. The processes in Figure 10(b) and (c) are repeated periodically. After T times of running these steps, the variables $y_{1,1}^1$ and $y_{1,2}^2$ are set to $x_{1,1}^1$ and $x_{1,2}^2$, respectively, and the updates by the users and helpers are repeated.

VIII. Simulation Results

We compare our proposed methods with a single layer VoD streaming using the helpers. For this purpose, we use LP optimization to find the optimal video placement and bandwidth allocation of the helpers to the users. We refer to this scheme as DIST methods. We also study the convergence of the proposed distributed method under the static and dynamic cases. For this purpose, we implemented a simulator in

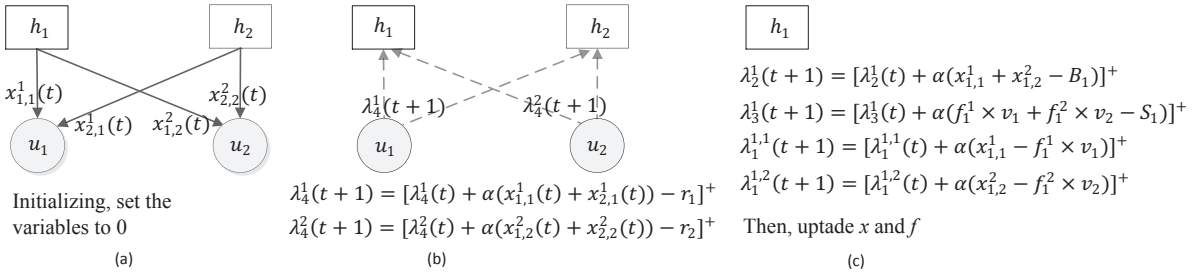
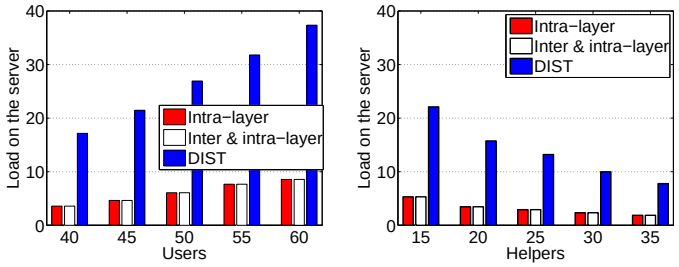


Fig. 10. Example of the distributed solution.

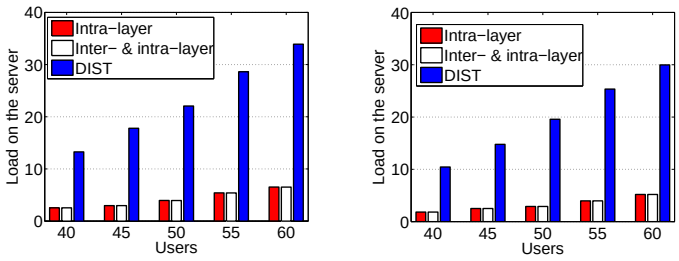
TABLE II
THE RANGES OF THE PARAMETERS IN THE SIMULATIONS.

Video's rate	Video's size	Bandwidth capacity	Storage capacity	Num. of adjacent helpers to a user
[1,2] kbps	[0.5,2] MB	[2,4] kbps	[0.5,2] MB	[1,3]



(a) Number of helpers: 20 (b) Number of users: 40

Fig. 11. Server's load in kbps, VoD, 5 videos, 5 layers.



(a) $a = 2$ (b) $a = 3$

Fig. 12. Server's load in kbps, VoD power-law distribution of the videos popularity, Number of helpers: 20, Number of layers: 5.

the Matlab environment. For solving the linear programming optimizations, we use the Linprog optimization toolbox which is embedded in Matlab.

We assume that the popularity of the videos and the number of subscribed layers by each user are uniformly distributed. We evaluate the methods on 100 random topologies, and use the average output of the simulation for plots of this section. By random topologies we mean randomly connecting the users to the helpers and randomly setting the bandwidth limit of the helpers. Number of adjacent helpers to user u_i is randomly selected in the range of [1,3]. Assuming that this number is d_i, we connect user u_i to d_i helpers. The range of a video's rate, size, storage capacity, bandwidth capacity, and number of adjacent helpers to each user are randomly chosen in the ranges shown in Table II.

A. Performance

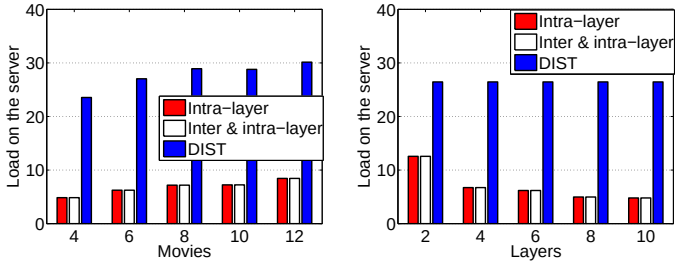
In Figure 11(a), we compare the loads on the central server. Each video contains 5 layers, and the number of requested layers by each user is randomly chosen in the range of $[1, 5]$. The other parameters are shown in Table II. The figure shows that the result of the joint inter- and intra-layer coding is very close to that of the intra-layer coding. Moreover, the server's load in our methods is up to 75% less than that of the DIST approach. The figure shows that the slope of the load in the DIST method is more than that of our proposed approaches, which is due to to users' greater amount of required resources when using the DIST method.

In our next experiment, we study the effect that the number of helpers has on the server's load. It is clear that more helpers can provide more portions of the videos, due to more available capacity and bandwidth resources. As a result, the server's load in all of the methods decreases as we increase the number of helpers, as illustrated in Figure 11(b).

We repeat the experiment in Figure 11(a) for the case that the popularity of the videos has a power-law distribution [34], i.e., the fraction of movies with d request (denoted as P_d) is proportional to d^{-a}: $P_d = (a-1)d^{-a}$. Here, a is the power-law distribution parameter, which usually satisfies $a \in [2, 3]$. In Figure 12(a), we set $a = 2$. The other parameters are the same as Figure 11(a). Comparing to the Figure 11(a), the load on the server reduces in Figure 12(a). The reason is that, in this case, the probability of common requests increases; thus, the storage of the helpers can be used more efficiently. We increase a to 3 and repeat the experiment. Figure 12(b) shows that as the popularity of few videos increase, the advantage of using the helpers increase as well; as a result, the load on the server reduces.

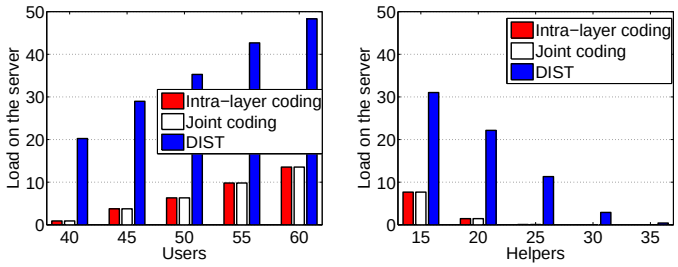
Figures 13(a) and (b) depict the effect of the number of videos and layers of the server's load. The simulation's parameters are chosen randomly in the ranges shown in Table II. The server's load of the methods increases as we increase the number of videos. This is because, as we increase the number of choices, the number of common requests decreases. As a result, the helpers need to store more videos, which is not feasible due to the storage limitations. More layers give the users the choice to select videos with a lower quality, which decreases the load on the server, as shown in Figure 13(b). In this figure, the server's load is almost fixed in the DIST method, since DIST is a no-layer approach.

In order to validate the result of Figure 11(a), we repeat the same experiment in the case of geographic distribution of the nodes. For this purpose, we distribute the nodes randomly in a 4×4 M area, and set the transmission range of the nodes to 1 M. Figure 14 (a) shows the load on the server of the different methods. In this figure, the video rates, video sizes, storage capacity, and bandwidth capacity of the helpers are in the range of $[1, 2]$, $[0.5, 2]$, $[0.5, 2]$, and $[1, 3]$, respectively. Most similar to Figure 11 (a), the load on the server increases as we increase



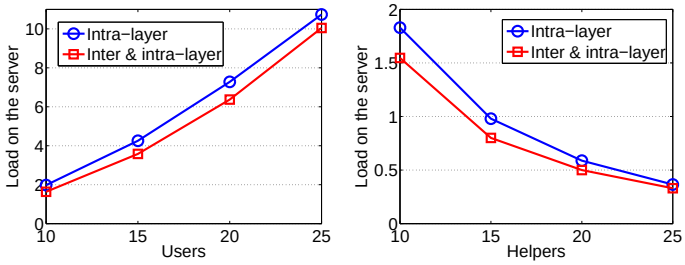
(a) Number of layers: 5 (b) Number of videos: 5

Fig. 13. Server's load in kbps, VoD, Number of users: 50, number of helpers: 20.



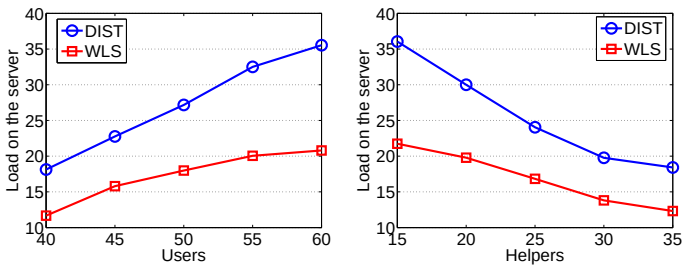
(a) Number of helpers: 20 (b) Number of users: 40

Fig. 14. A server's load (in kbps) in the case of geographic distribution of the user and helpers, VoD, Number of videos: 5, number of layers: 5.



(a) Number of helpers: 10 (b) Number of users: 10

Fig. 15. The advantage of joint inter- & intra-layer coding over intra-layer coding, VoD, Number of layers: 4.



(a) Number of helpers: 20 (b) Number of users: 50

Fig. 16. Live streaming, Load on the server in kbps, Number of videos: 5, number of layers: 1.

the number of users, which is due to more video requests. In Figure 14(a), the average number of adjacent helpers to the users is equal to 4.2. Figure 14(b) shows the effect of changing the number of helpers on the server's load. As it is expected, most similar to Figure 11(b), the load on the server decreases as we increase the number of helpers.

As we stated in the introduction, there are cases where the inter-layer NC reduces the server's load. However, Figures 11 and 13 show that the server's load using joint NC and just intra-layer coding are very close. In order to study the benefit of inter-layer coding, we repeat the first experiment with a single video, as to eliminate competition between the users with different video requests. The helpers' bandwidths are in the range of $[5, 10]$, and the video size, video rate, and the storage capacities are set to 4, 4, and 1. Also, the degree of each user is in the range of $[1, 4]$, and we set the number of requested layers of each user to its degree. Figure 15(a) shows that the server's load using joint coding is up to 17% less than that of the intra-layer coding method. Figure 15(b) shows that, as we increase the number of helpers, the difference between the methods decreases, which is due to the availability of a high percentage of the video through the helpers in both methods. Based on our observation, we can find that when the users compete to receive different videos, and the bandwidth is the bottleneck, inter-layer NC cannot increase the content available to the users. As a result, we can conclude inter-layer NC is not much useful in practice.

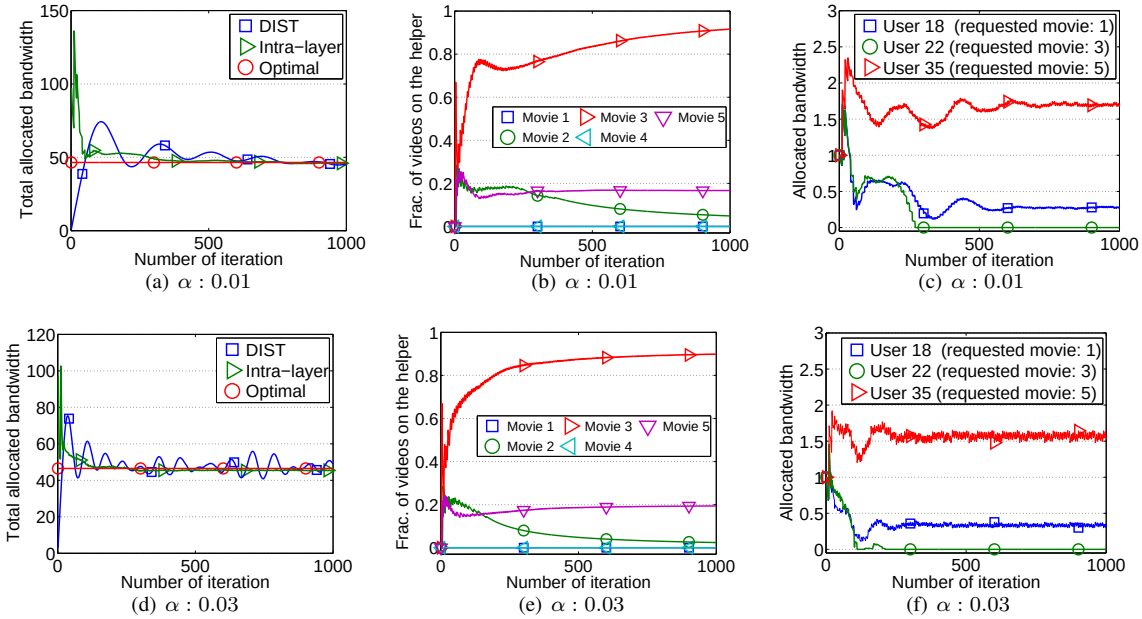
Figures 16(a) and (b) show the comparison between the server's load in the DIST and WLS (wireless live streaming) methods. The experimental parameters are chosen randomly in the ranges shown in Table II. In the case of LS, the playback time of the users that watch the same video are synchronous. Thus, in the WLS method, the helpers do not assign a separate bandwidth to the users that watch the same video, which results in providing more portions of the videos through the helpers. As a result, the server load in the WLS method is less than that of the DIST method. In Figure 16(a), the slope of DIST is more than that of the WLS, which means that the helpers do not have enough free bandwidth to support more users. On the other hand, in Figure 16(b), WLS has less slope than the DIST method since, even in the case of 15 helpers, the users receive a large portion of their requests.

B. Convergence

In this section, we study the convergence of our distributed solution under both the static and dynamic cases.

1) Static System: We evaluate the convergence of the proposed distributed algorithm in Figure 17. We solve the DIST method using a similar scheme to our solution in the case of inter-layer coding. However, in order to study the effect of Proximal method, we do not use proximal method for DIST. In this figure, the number of users, helpers, and videos are equal to 50, 20, and 5, respectively. In order to have a fair comparison, we set the number of video layers to 1. The optimal solution is computed off-line for comparison. It is clear in Figure 17(a) that the proposed distributed solution converges to the optimal solution very fast; however, the convergence speed of the DIST approach is less than that of our approach. Moreover, the DIST method oscillates around the optimal solution. Figure 17(b) depicts the convergence of a particular helper's (helper h_5) storage allocation. The allocated storage for videos 2 and 4 goes to zero, since these videos are not requested by the adjacent users of this helper. The convergence of the allocated bandwidth from helper h_5 to its adjacent users is shown in Figure 17(c).

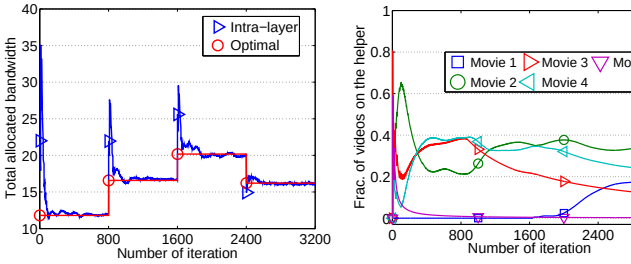
We repeat the previous experiment by increasing the step size α from 0.01 to 0.03. The results are shown in Figures 17(d), (e), and (f). By comparing Figures 17(a) and (d), it can be inferred that our distributed method converges faster to the optimal solution as we increase the step size. Moreover,



(a) α : 0.01 (b) α : 0.01 (c) α : 0.01

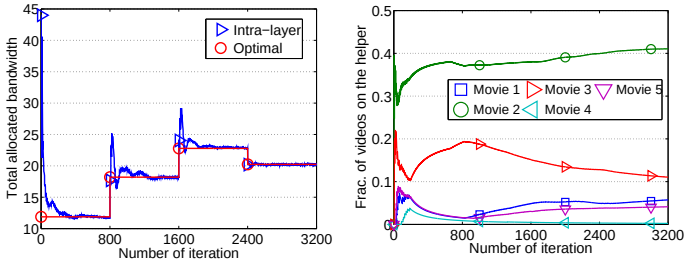
(d) α : 0.03 (e) α : 0.03 (f) α : 0.03

Fig. 17. VoD with intra-layer coding. Convergence of the proposed distributed method to the optimal solution in a static network case. The number of users, helpers, and videos are equal to 50, 20, and 5, respectively. (a) and (d): Total allocated bandwidth to the users. (b) and (e): The fraction of each video on helper h_5. (c) and (f): The allocated bandwidth from helper h_5 to its adjacent users.



(a) (b)

Fig. 18. Convergence of the proposed distributed method to the optimal solution in the case of dynamic users, Number of helpers: 10, Number of videos: 5. (a) Total allocated bandwidth. (b) Frac. of videos on helper h_8.



(a) (b)

Fig. 19. Convergence of the proposed distributed method to the optimal solution in the case of dynamic helpers, Number of users: 20, Number of videos: 5. (a) Total allocated bandwidth. (b) Frac. of videos on helper h_3.

even with a greater α, our method does not oscillate. On the other hand, the DIST method's oscillation increases rapidly as we increase the step size. Figures 17(e) and (f) illustrate the bandwidth and storage allocation of helper h_5, respectively.

2) Dynamic System: In this section, we show that our distributed approach automatically adapts to the system dynamics. As a result, the users and the helpers only need to run the distributed algorithm, regardless of the changes in the system.

We study the effect of changing the number of users to the system in Figure 18. For this purpose, we add 5 users at both iterations 800 and 1600, and we randomly connect them to [1,3] helpers. We also remove 5 users at iteration 2400. The initial number of users is 10, and there are 10 helpers in the system. We set the number of videos to 5. The optimal solution is computed off-line for comparison. Figure 18(a) shows that the total allocated bandwidth of the optimal solution changes as we add or remove users, and the distributed solution converges to the optimal result. We depict the fraction of stored videos on a helper h_8 in Figure 18(b).

We repeat the previous simulation for the case of dynamic helpers. We set the number of users, helpers, and videos to 20, 6, and 5, respectively. We add 3 new helpers at iterations 800 and 1600, and remove 3 helpers at iteration 2400. Figures 19(a) and (b) show that the proposed distributed method adapts to the changes in the dynamic case. After removing 3 helpers, the the allocated bandwidth does not return to the level of iteration 1600. The reason is that, the removed helpers are not those that are added at iteration 1600. We can conclude that the storage size or bandwidth limit of the removed helpers are less than those of the added helpers. Another possibility is that the added helpers covered the users with a common video request, but the removed helpers did not.

C. Unreliable links

Here, we repeat the experiment in Section VIII-A for the topologies with unreliable links between the helpers and the users. We set the reliability of these links in the range of $\epsilon \in [0.8, 1]$, and measure the effect of number of users on the server load. The number of helpers, movies, and the video layers are equal to 20, 5, and 5 respectively. The server load of the methods increases in Figure 20(a) as we increase the number of users, which is due to limited bandwidth and storage resources. By comparing Figures 20(a) and 11(a), we find that the gap between our proposed method and the DIST

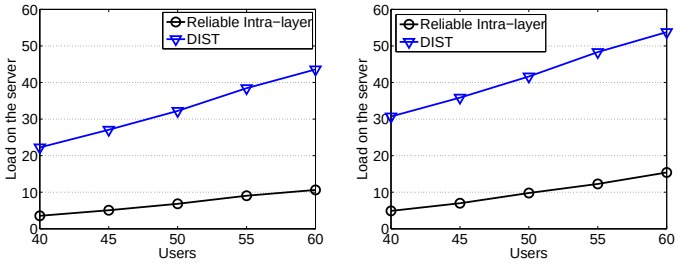


Fig. 20. Server's load in kbps, in the case of unreliable links from helpers to the users and reliable server links, VoD, Number of helpers: 20, Number of videos: 5, number of layers: 5. (a) $\epsilon \in [0.8, 1]$ (b): $\epsilon \in [0.6, 0.8]$

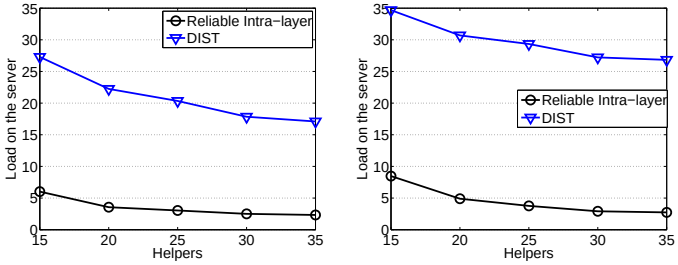


Fig. 21. Server's load in kbps, in the case of unreliable links from helpers to the users and reliable server links, VoD, Number of users: 40, Number of videos: 5, number of layers: 5. (a) $\epsilon \in [0.8, 1]$ (b): $\epsilon \in [0.6, 0.8]$

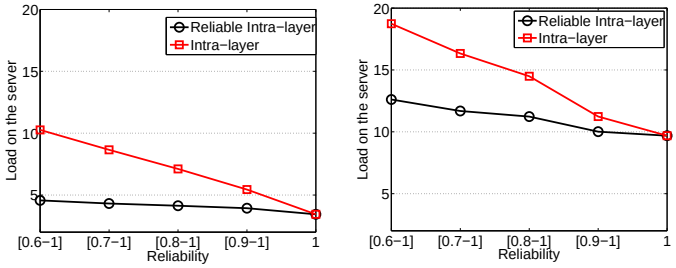


Fig. 22. Server's load in the case of unreliable links from helpers to the users and reliable server links, VoD, Number of helpers: 20, number of videos: 5, number of layers: 5.

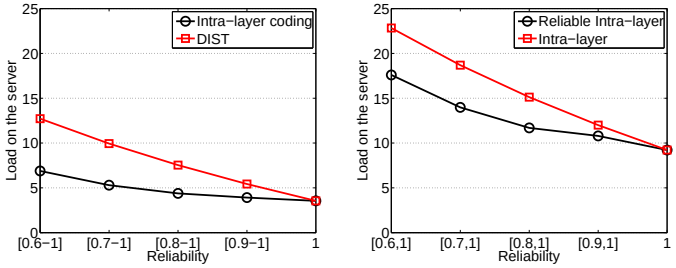


Fig. 23. Server's load in kbps, in the case of unreliable links from helpers and the server to the users, VoD. Number of helpers: 20, number of videos: 5, number of layers: 5.

method is more in Figure 20(a). This is because the DIST method does not consider the unreliability of the links.

In Figure 20(b), we reduce the reliability of the links to the range of $\epsilon \in [0.6, 0.8]$ and repeat the previous experiment. The two reasons that make the server load in the DIST method more than the reliable intra-layer coding are a.) the layered approach in our method, and b.) considering the reliability of the links. Having more users increases the requests for the resources, which results in more load on the server. The server load in Figure 20(b) is more that that of in Figure 20(a), which is due to less reliable helpers' links.

The number of users, videos, and layers in Figures 21(a) and (b) are equal to 40, 5, and 5, respectively. We set the range of links' reliability in Figure 21(a) to $\epsilon \in [0.8, 1]$. Most similar to Figure 13(a), the load on the sever decreases as we increase the number of helpers. The reason is that, more helpers can provide a larger portion of the videos to the users. We change the reliability of the links from $\epsilon \in [0.8, 1]$ to $\epsilon \in [0.6, 0.8]$, and repeat the previous experiment. By comparing Figures 21(a) and (b), we find that the unreliability of the links has more of a negative effect on the DIST approach, compared to the reliable intra-layer coding method.

Next, we compare the proposed intra-layer coding method with the reliable intra-layer coding. Figure 22(a) shows the load on the server for different ranges of link reliability. The number of helpers, videos, and layers are equal to 20, 5, and 5, respectively. Moreover, we set the number of users to 40. As expected, the proposed reliable intra-layer coding method reduces the load on the server, since it considers the unreliable links between the helpers and the users. This method gives more priority to the more reliable helper links, since giving the same priority to the links results in wasting the bandwidth resources. As the reliability of the links increases, the gap between the proposed method vanishes. In the case of reliable links, the server load of the methods becomes the same. We repeat the same experiment for 60 users in Figure 22(b), which results in more server load that that of in Figure 22(a).

We repeat the last two experiments in the case of unreliable links from the server and helpers to the users. Obviously, the server lossy links result in more of a load on the server, which can also be inferred from Figures 23(a) and (b). Remember that, in the case of lossy server links to the users, we change the objective function from maximizing the provided help through the helpers to minimizing the load on the server.

IX. Conclusion

In this paper, we study the problem of utilizing helpers to minimize the load on the central video servers. For this purpose, we formulate the problem as an LP optimization problem. This is done by using joint inter- and intra-layer NC. We discuss the advantages of joint inter- and intra-layer NC over just intra-layer NC, and through an empirical study, we found the cases in which joint coding reduces the server's load. We use a lightweight triangular inter-layer NC instead of the general form of inter-layer NC, to reduce the time complexity of the optimization. We solve the proposed optimization in a distributed way, and evaluate the convergence and the gain of our distributed approach via comprehensive simulations. Our future work is to consider the cost of helpers in the optimization and study the overhead that results from introducing the helpers.

References

[1] A. Finamore, M. Mellia, M. Munafò, R. Torres, and S. Rao, "Youtube everywhere: impact of device and infrastructure synergies on user experience," in *ACM IMC*, 2011, pp. 345–360.

[2] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," in *ACM SIGCOMM*, 2010, pp. 75–86.

[3] S. Pawar, S. Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a distributed caching based video-on-demand system," in *ACSSC*, 2011.

[4] H. Hao, M. Chen, A. Parekh, and K. Ramchandran, "A distributed multichannel demand-adaptive P2P VoD system with optimized caching and neighbor-selection," in *SPIE*, 2011.

[5] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," *Arxiv preprint arXiv:1109.4179*, 2011.

[6] J. Wang and K. Ramchandran, "Enhancing peer-to-peer live multicast quality using helpers," in *IEEE ICIP*, 2008, pp. 2300–2303.

[7] H. Zhang, J. Wang, M. Chen, and K. Ramchandran, "Scaling peer-to-peer video-on-demand systems using helpers," in *IEEE ICIP*, 2009, pp. 3053–3056.

[8] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM CCR*, 1996, pp. 117–130.

[9] M. Kim, D. Lucani, X. Shi, F. Zhao, and M. Médard, "Network coding for multi-resolution multicast," in *IEEE INFOCOM*, 2010, pp. 1–9.

[10] N. Shacham, "Multipoint communication by hierarchically encoded data," in *IEEE INFOCOM*, 1992, pp. 2107–2114.

[11] M. Effros, "Universal multiresolution source codes," *IEEE Transactions on Information Theory*, vol. 47, no. 6, pp. 2113–2129, 2001.

[12] T. Fang and L. Chau, "Op-based channel rate allocation using genetic algorithm for scalable video streaming over error-prone networks," *Image Processing, IEEE Transactions on*, vol. 15, no. 6, pp. 71 323–1330, 2006.

[13] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[14] M. Shao, S. Dumitrescu, and X. Wu, "Layered multicast with inter-layer network coding for multimedia streaming," *IEEE Transactions on Multimedia*, vol. 13, no. 99, pp. 353–365, 2011.

[15] 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.

[16] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[17] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[18] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE Transactions on Information Theory*, vol. 11, no. 5, pp. 782–795, 2003.

[19] P. Ostovari, J. Wu, and A. Khreishah, "Network coding techniques for wireless and sensor networks," in *The Art of Wireless Sensor Networks*, H. M. Ammari, Ed. Springer, 2013.

[20] B. Li, Z. Wang, J. Liu, and W. Zhu, "Two decades of internet video streaming: A retrospective view," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1s, pp. 1–20, 2013.

[21] T. Ho, M. Médard, R. Koetter, D. 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.

[22] J. Wang, C. Huang, and J. Li, "On ISP-friendly rate allocation for peer-assisted vod," in *ACM Multimedia*, 2008, pp. 279–288.

[23] C. Wu and B. Li, "On meeting P2P streaming bandwidth demand with limited supplies," in *SPIE MMCN*, 2008.

[24] J. Wang, C. Huang, and J. Li, "Challenges, design and analysis of a large-scale P2P-vod system," in *ACM SIGCOMM*, 2008, pp. 375–388.

[25] M. Luby, "LT codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–280.

[26] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[27] Y. He and L. Guan, "A new polynomial-time algorithm for linear programming," in *ACM STOC*, 1984, pp. 302–311.

[28] D. Koutsonikolas, Y. Hu, C. Wang, M. Comer, and A. Mohamed, "Efficient online wifi delivery of layered-coding media using inter-layer network coding," in *IEEE ICDCS*, 2011, pp. 237–247.

[29] E. Chong and S. Zak, *An Introduction to Optimization*. John Wiley & Sons, 2013.

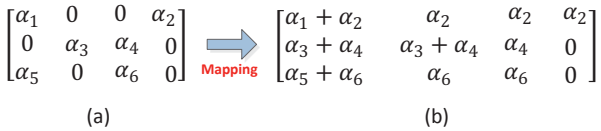


Fig. 24. (a) Coefficient matrix of a set of general linear coded packets. (b) Coefficient matrix of a set of triangular coded packets.

[30] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ (USA); Prentice Hall Inc., 1989.

[31] X. Lin and N. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Transactions on Automatic Control*, vol. 5, no. 51, pp. 766–781, 2006.

[32] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ Press, 2004.

[33] A. Khreishah, C.-C. Wang, and N. B. Shroff, "Optimization based rate control for communication networks with inter-session network coding," in *IEEE INFOCOM*, 2008, pp. 81–85.

[34] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[35] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, 4th ed. Upper Saddle River, NJ (USA); Prentice Hall, 2003.

APPENDIX A
PROOF OF LEMMA 1

Assume that A is the coefficient matrix of a set of general linear coded packets. Each row of A represents the coefficients of a coded packet. Consider d_i as the rightmost non-zero column in row i. In order to map row i to a triangular form, we should change it in way that the columns 1 to d_i becomes non-zero. The rank of a matrix is preserved under elementary column operations [35]. Therefore, we can multiply a column by a number and add the result to another column without changing the rank of the matrix.

Our mapping works as follows. We start from the rightmost column j and add it to its left column $j-1$. Then, we repeat this process by adding the new values of column $j-1$ to column $j-2$. This mapping is done for all of the columns. At the end, matrix A is converted to a coefficients matrix A', which contains the coefficients of a set of triangular coded packets. It should be noted that even in the case that adding column k to column $k-1$ results to a zero value for some of the cells in column $k-1$, we can multiply column k by a number before adding it to column $k-1$ to prevent the zero cells to from. Figures 24(a) and (b) show the coefficient matrices of a set of general linear coded packets and their mapping to a set of triangular coded packets.

APPENDIX B
PROOF OF LEMMA 2

Clearly, receiving more than n_1 coded packets from layer 1 is not useful to user u, since n_1 coded packets are enough to decode layer 1. In other words, receiving n_1 linearly independent coded packets results in a rank equal to n_1, and receiving more coded packets does not increase the rank. Coded layer 2 contains coded packets over the first two original layers. As a result, the user does not need more than n_1+n_2 coded packets from the first two layers. With the same reasoning, receiving more than $\sum_{l=1}^{l'} n_l$ coded packets from the first l' coded layers is not useful.