J. Vis. Commun. Image R. 43 (2017) 10–20

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Adaptive packet scheduling for scalable video streaming with network coding ☆

Shenglan Huang *, Ebroul Izquierdo, Pengwei Hao

Queen Mary, University of London, United Kingdom

ARTICLE INFO

Article history:
Received 20 April 2016
Revised 25 September 2016
Accepted 20 November 2016
Available online 30 November 2016

Keywords:
Network coding
Scalable video streaming
Live streaming
P2P network

ABSTRACT

Over the last decade, the emergence of new multimedia devices has motivated the research on efficient media streaming mechanisms that adapt to dynamic network conditions and heterogeneous devices' capabilities. Network coding as a rateless code has been applied to collaborative media streaming applications and brings substantial improvements regarding throughput and delay. However, little attention has been given to the recoverability of encoded data, especially for the streaming with a strict deadline. This in turn leads to severe quality of experience. In this paper, we solve the unrecoverable transmission by proposing a multi-generation packet scheduling problem, which is treated as a video quality maximization problem and solved using dynamic programming algorithm. Experimental results confirm that the proposed algorithm brings better data recoverability and better quality of service in terms of video quality, delivery ratio, lower redundancy rate under different network sizes.

1. Introduction

With the emergence of various multimedia devices such as laptops of smartphones, the scalable data streaming mechanisms have gained great popularity among users. Network users spend significant time in online communication sharing media and watching videos streamed over the Internet. Indeed, multimedia communications are nowadays pervasive and substantial part of modern life. Many research has been studied in the creation of a natural multimedia environment for remote immersive and interpersonal communication, like high-quality 3D video coding or HEVC. With high-quality 3D virtual world and 3D videos, the efficient transmission of data is also the key to success. To achieve the efficient transmission, some researchers combine scalable video coding with network coding [1,2] to adaptively optimize the video transmission over P2P network. Many researchers also study in multi-source streaming tree technologies. Comparatively, little attention has been given to improving bandwidth efficiency through suitable packet scheduling schemes. However, improving bandwidth efficiency is necessary because better bandwidth resources allocation leads to more useful transmissions and better quality of service for media streaming applications. Therefore, this paper focuses on improving bandwidth efficiency in scalable streaming networks.

Methods such as Fountain codes (Random Linear Network Coding [3], Raptor codes [4], LT codes [5]) have been proposed recently for efficient data delivery in the lossy network. The benefits of Fountain codes is the improvement in delay reduction for live media streaming applications. The traditional scheme for transferring data across an erasure channel is a continuous two-way communication, where senders acquire acknowledgments from receivers to re-transmit error packets. With Fountain Codes, senders can keep transmitting encoded packets to receivers until receivers have enough valid packets to decode the original packets. Raptor codes and LT codes are based on the exclusive or operation and need the complete original data to generate new coded data. Comparatively, RLNC linearly combines packets with random coefficients [6] and can generate new coded data from part of original packets. Therefore, RLNC is more suitable to be used in a distributed system than Raptor codes and LT code. Based on RLNC, Chou et al. [7] proposed a practical network coding scheme for large-scale media streaming. The authors proved that practical network coding could provide significant gains regarding throughput and delay compared with the traditional approach in a distributed network. Many large-scale applications of applying network coding in the field of multimedia streaming [8–10] have also demonstrated benefits in reducing communication delays and facilitating collaboration among nodes. Further, with the development of scalable video coding, some scalable coding methods are combined with RLNC to provide unequal protection for scalable data transmission, such as Expanding Foutain codes [11], Hierarchical network coding [12], and Layered network coding [13].

☆ This paper has been recommended for acceptance by M.T. Sun.

* Corresponding author.
E-mail address: s.huang@qmul.ac.uk (S. Huang).

http://dx.doi.org/10.1016/j.jvcir.2016.11.014
1047-3203/

Nomenclature

$\mathcal{N}_i$	network node i, $i \in [0, N]$	α	independent threshold for a receiver node to become a sender node
A_j	neighborhood of the receiver node $\mathcal{N}_j$	Γ	priority region, $\Gamma = [\Gamma_s, \ldots, \Gamma_e]$
g	temporal index (generation) in the media, $g \in [1, G]$	ω	urgent region, $\omega = [\omega_s, \ldots, \omega_e]$
κ	duration of each generation	l	layer index in the media, $l \in [1, L]$
V	size of video packets	$s_{(g,l)}$	number of video packets in generation g, layer l
U_i	upload bandwidth of streaming node $\mathcal{N}_i$, $i \in [1, N]$ [kByte/s]	$s_{(\hat{g},l)}$	number of actually scheduled packets in generation g, layer l
U_{ij}	upload bandwidth allocated from $\mathcal{N}_i$ to $\mathcal{N}_j$ [kByte/s]	$Q_{(g,l)}$	number of quality gain of generation g, layer l
$\widehat{U}_j$	overall available upload bandwidth from all senders to the receiver node $\mathcal{N}_j$ [kByte/s]	μ	the unsuccessful transmission rate
$\overline{U}_j$	the number of packet can be received from A_j in each generation [packets/κ s]	d_{ij}	number of scheduled packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ in generation g
$\overline{U}$	abbreviation of $\overline{U}_j$	$t_{(g,l)}$	denotes whether generation g and layer l is subscribed by the receiver, $t_{(g,l)} \in (0, 1)$.
S	average streaming rate		

Peer-to-peer (P2P) networks have been used extensively in multimedia streaming as an effective transmission platform. Many existing commercial P2P networks have shown their success when they are used to transmit files [14] or multicast streaming [15,16]. The advantage of P2P systems is that the network construction is scalable and money-saving [17–19]. Early P2P networks are based on multicast trees. Peers immediately push packets to its children nodes when they receive new data. In contrast, swarm-based schemes are more amenable to real-world implementations [20]. In the *swarm-pull* scheme each peer maintains its neighbors and periodically exchanges its *buffer-map*, which represents the availability of video blocks, with its neighbors. Each peer then fetches packets from its neighbors accordingly [21,16]. By taking advantage of network coding, the *mesh-push* schemes are proposed [22,23,1]. They allow peers in *swarm-push* schemes actively push their received packets to its neighbors according to periodically exchanged *buffer-maps*. When a receiver gets enough encoded packets, it immediately decodes the original packets using Gaussian elimination.

Despite substantial progress in the field, little attention has been given to bandwidth efficiency in the NC-based streaming applications, especially to the NC-based scalable video streaming. Better bandwidth efficiency will result in a better quality of service in multimedia streaming. The bandwidth inefficiency in current NC-based *mesh-push* scheme is caused by the delay of *buffer-map*. In the push scheme, there will be a good number of redundant packets already in the transmission pipeline if the delay is considerable. We term this kind of redundant transmission as *braking effect*. The *braking effect* is especially severe in scalable video streaming because it happens very frequently due to the layered transmission mechanism. To solve this problem, instead of transmitting layered data, the optimal layer is selected in advance based on the information of video, network bandwidth, and loss rate in advance. Senders then push packets to each receiver based on a distributed multi-sender cooperation algorithm.

The remainder of this paper is organized as follows. In Section 2 we introduce some related works on the packet scheduling problem. In Section 3, we propose the overall system model for the streaming network. Then in Section 4 we describe the proposed scalable streaming system. In Section 7 we present results of some simulation experiments, and in Section 8 we draw some conclusions.

2. Related work

Scalable media streaming refers to the coding technique which fragments a single high-quality media-stream to several layers to provide different quality of experience for different devices. Some papers have proposed the use of NC with SVC. For example, the authors in [12] combines the lower layer first scheduling policy with hierarchical random push network coding. In [24], Nguyen et al. proposed to use a drop-threshold and an add-threshold to find the suitable layer to subscribe based on the *buffer-map* of receivers. Compared with [12] and the random approach, the *uninformative* packets rate is relatively low. However, the layer subscription in [12] is fluctuant and inaccurate. In [2], Sanna pointed out the delay of the *buffer-map* updating will generate braking effect, and brings superfluous transmission. To solve this problem, in [2], they proposed a bandwidth estimation and proactive rate selection algorithm as a plugin component for the random-push packet scheduling policy. In [25], Thomos et al. proposed a rate allocation method to maximize the received video quality. In their scheme, they formulate the bandwidth allocation problem as a distortion optimization function. After that, Huang et al. [23] proposed a joint video quality and delay optimization function. However, both of their works optimize the packet scheduling based on the single generation. Such scheduling policies cannot fully integrate network resources, and maximize the bandwidth efficiencies. Another previous work [26] constructs the cost function for each possible transmitting policy. Then the sender pushes packets according to the rank of cost. Nevertheless, it still cannot avoid the bandwidth inefficiencies caused by these *unrecoverable* packets. In [27], Thomos proposes to use the Markov decision processes and reinforcement learning approaches to find the possible optimal pushing strategy for a given scheduling region. However, the slow convergence speed and high computational complexity make it less competitive.

Our work departs from the conventional choice of sequential processing of generations and considers the scheduling of several generations simultaneously. The advantage is that the limited bandwidth can be more accurately allocated to multiple generations and layers such that the maximum bandwidth efficiency can be achieved. To achieve the accurate bandwidth scheduling, a multiple generation scheduling problem is proposed firstly, which has been proved to be an NP-complete problem. To solve this problem, we proposed two methods. One algorithm transfers the multiple generation scheduling problem to single generation scheduling problem. The other algorithm solves the multi-generation optimization using a novel dynamic programming algorithm, which can be solved in pseudo polynomial-time. Based on the selected classes, a distributed packet scheduling algorithm coordinates senders to distribute packets. The scheduling algorithm can effectively avoid the transmission of unrecoverable data, thereby improving the bandwidth efficiency and QoS.

3. System model

In this section, a brief overview of the system is given. Firstly, the definitions of the transmission network are given, and the

construction process of the transmission network is briefly explained. Secondly, the model for scalable media data is explained to understand the scheduling process and coding process of scalable data.

3.1. Network model

The overall network model is depicted in Fig. 1. In this model, three types of peers are defined: *streaming source*, *tracker server*, and *client node*. Each peer contacts the tracker server to join the network. Some control information, i.e., the upload bandwidth of neighboring nodes and the size of video packets, is exchanged during the procedure. The overlay of network nodes is represented as a graph $(\mathcal{N}, \varepsilon)$ composed for nodes $\mathcal{N} = \{\mathcal{N}_0, \ldots, \mathcal{N}_N\}$ and the edge ε. $\mathcal{N}_0$ represents the *tracker server* and $\mathcal{N}_1$ represents the *streaming source*. The rest nodes from $\mathcal{N}_2$ to $\mathcal{N}_N$ are client nodes. A client node becomes a sender node when it holds at least α linear independent segments $(0 \leqslant \alpha \leqslant 1)$. $A_j \subset \mathcal{N}$ is defined as the neighborhood of $\mathcal{N}_j$, which is the set of sender nodes that are connected to $\mathcal{N}_j$.

For any nodes in the network, $U = \{U_0, \ldots, U_N\}$ are indicated as the vector of streaming nodes. Each sender $\mathcal{N}_i$ equally allocates its upload bandwidth U_i to its receiving nodes $\mathcal{N}_j$, and the allocated upload bandwidth from node $\mathcal{N}_i$ to $\mathcal{N}_j$ is U_{ij} such that $\sum_{i=0}^{|A_j|} U_{ij} = U_j$. In this way, the low bandwidth of some nodes won't become the bandwidth bottleneck of the whole network. The overall upload bandwidth which allocated to $\mathcal{N}_j$ is $\widehat{U}_j = \sum_{i=1}^{|A_j|} U_{ij}$. To simplify the discussion, we also define the available κ-quantized upload bandwidth of $\mathcal{N}_j$ is $\overline{U}_j = \widehat{U}_j * \kappa / V$. As the scheduling algorithm is a distributed packet scheduling algorithm and designed for each receiver node $\mathcal{N}_j$, $\overline{U}$ is used to represent $\overline{U}_j$ for simplicity.

3.2. Media streaming model

The scalable media stream which is distributed to the network nodes is modeled as a bi-dimensional array of generations shown in Fig. 2. A generation is made up by one or several group of pictures (GOP) in the media. Each generation is identified within the media stream by a temporal index $g \in [1, G]$. Each generation with identical temporal index g has the same duration κ. Each generation g is subdivided into several layers $l \in [1, L]$. In the following section, the notation (g, l) is used to indicate the **class** with a temporal index g and layer index l. Each class (g, l) has $s_{(g,l)}$ blocks. $Q(g, l)$ is the quality enhancement of each class (g, l). Due to the dependency among layers, the lower layer needed to be decoded first before the higher layer can be decoded. When the highest layer L is decoded, the quality gain of generation g is $\sum_{l=0}^{L} Q(g, l)$.

Instead of transmitting raw video packets, the P2P network transmits network-coded packets. A new coded packet is generated by combining the blocks belonging to one generation together with random coefficients. The algebraic operations are performed in Galois field F_q (usually $q = 2^8$). In this system, the blocks are encoded from the base layer to the subscribed layer l together.

$$y = \sum_{i=1}^{s_{(g,1)}+s_{(g,2)}+\ldots+s_{(g,l)}} b_i c_i \tag{1}$$

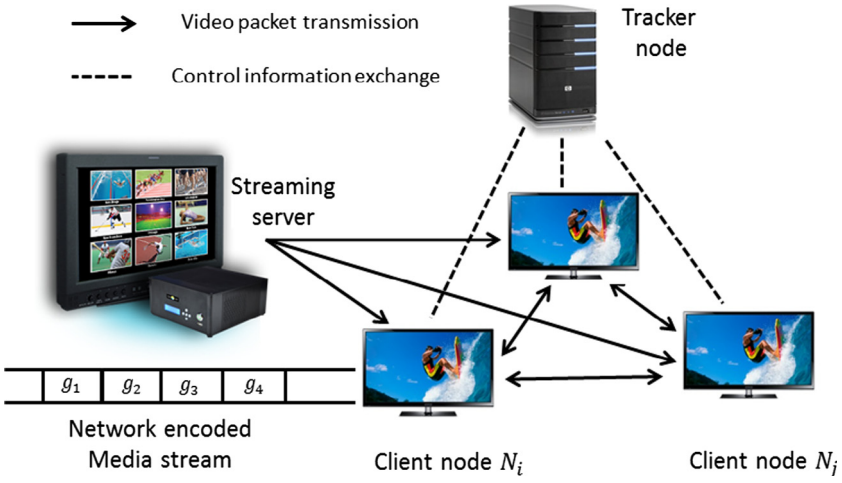

Fig. 1. The overall system model.

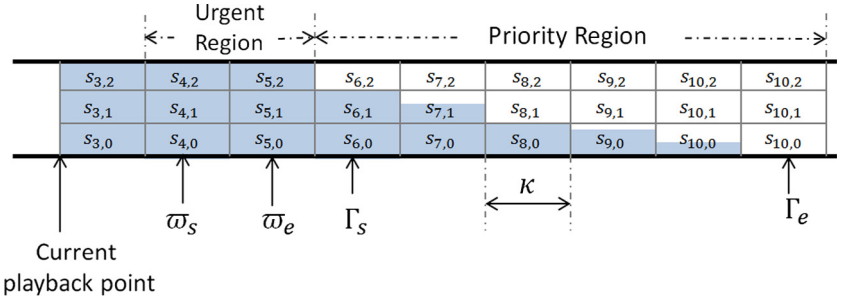

Fig. 2. Sample media stream model for scalable-coded media stream.

In the Eq. (1), $s_{(g,l)}$ is the number of packets in generation g of layer l, b_i is the ith raw blocks in generation g and c_i is a random coding coefficient. When a receiver receives y, it checks if the packet is *informative*. If the packet is *uninformative*, it is discarded. Once a client node has collected enough informative packets in a generation, it will recover this generation using the progressive Gauss-Jordan elimination.

To achieve a guaranteed transmission, the whole *transmission region* can be further divided into *urgent region* and *priority region*. In Fig. 2, $g = 3$ is the playback point. The urgent region ω is a sliding window next to the playback point with fixed size ω_l. The priority region Γ is a sliding window next to the urgent region with fixed size Γ_l. The start point and the end point of the priority region is defined as Γ_s and Γ_e respectively.

In the *priority region*, the data transmission follows the multiple generation scheduling algorithm proposed in Section 4 to achieve an optimized system efficiency. In the *urgent region*, transmission follows the request model proposed in Section 6 to further improve the delivery ratio. This design can make sure a reliable system performance when the network bandwidth flutters. Any unrecoverable generations out of the transmission region are discarded automatically.

3.3. Overall of transmission process

The overall of the video data transmission process is summarized in Fig. 3. First, each receiver contacts with the tracker node to obtain the video megedata (number of packets in each layer and each generation), link information (the available upload bandwidth U_{ij} between the sender node $\mathcal{N}_i$, and the receiver node $\mathcal{N}_j$ and the unsuccessful transmission rate μ). Then the receiver node calculates the compensated data parameter according to Scheduling Compensation Model (SCM) proposed in [28]. The main function of SCM is to calculate the number of packets $s_{(\hat{g},l)}$ that need to be sent from $\mathcal{N}_i \in A_j$ to $\mathcal{N}_j$ such that $s_{(g,l)}$ independent packets can successfully arrive at $\mathcal{N}_j$. After this, each receiver determines its subscribed layer l for each generation g in the *transmission region* based on the multiple-generation scheduling algorithm (MGS) and single-generation scheduling algorithm (SGS) proposed in Section 4. Subsequently, according to the request of each receiver, senders cooperatively transmit the selected class (g,l) to the receiver according to the distributed packet scheduling algorithm proposed in Section 5.

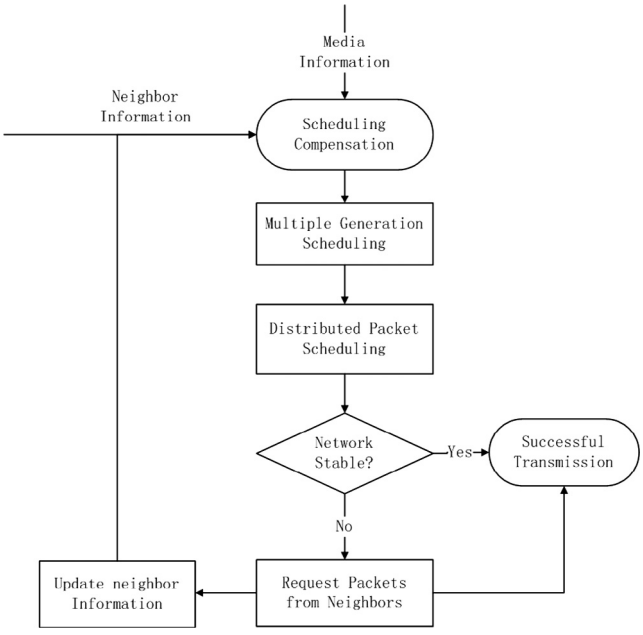

Fig. 3. Illustration of the overall transmission procedure at each client node.

When the network is stable, the subscribed class (g,l) can be successfully transmitted to the receiver. When the network is unstable, the subscribed class (g,l) may not be decoded in the priority region. Then when the class (g,l) is in the *urgent region*, the receiver will actively request packets from its neighbor nodes according to the request model proposed in Section 6. The *urgent region* and the *priority region* moves as the playback point moves.

4. Multiple generation scheduling

This section proposes a multiple-generation scheduling optimization problem to make fully use of the bandwidth resources in a given region. It determines which class should be subscribed and transmitted so that the overall video quality in the priority region can be maximized. The problem is formulated as a multi-generation PSNR maximization problem, and the optimal class subscription policy can be found through solving the maximization problem under some bandwidth constraints. To solve the scheduling problem, two algorithms are proposed. First, the problem is reformulated and solved through a single generation scheduling algorithm. Second, the problem is solved directly through a dynamic programming algorithm. As the proposed optimization problem is a variation of the Multiple-Choice Knapsack Problem (MCKP) [29]. Although the MCKP problem has been proved as an NP-complete problem in [30], the optimal solution can be get in pseudo polynomial-time through the dynamic programming algorithm [31]. Therefore, according to the classic dynamic programming for MCKP, we solved the multiple generation scheduling problem using the dynamic programming algorithm.

4.1. Problem formulation

In this section, a multiple-generation scheduling problem is proposed to find the most suitable class subscription policy. This problem finds the optimal layer subscription policy for each generation to maximize the received video quality, which can also reduce the uninformative and unrecoverable transmission. We find the optimal class subscription strategy by proposing a video quality maximization problem under the fully video packet recovery constraint. It chooses a suitable layer for each receiver to subscribe based on the estimated bandwidth $\overline{U}$. The multiple generation scheduling is performed in the priority region Γ. The optimization problem is shown in Eq. (2).

For any generation g, and any layer l in the region, $t_{(g,l)} \in (0,1)$ denotes whether generation g and layer l is subscribed by the receiver. The $s_{(\hat{g},l)}$ is used to represent the actually needed number of packets to transmit the class (g,l) after considering the scheduling compensation model proposed in [28]. The scheduling algorithm is computed every Γ_l.

$$t^*_{(g,l)} = \arg\max_{t_{(g,l)}} \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{l=0}^{L} Q_{(g,l)} t_{(g,l)}$$

$$\text{subject to} \sum_{g=\Gamma_s}^{k} \sum_{l=0}^{L} t_{(g,l)} s_{(\hat{g},l)} \leqslant \sum_{g=\Gamma_s}^{k} \overline{U}_g \text{ for } \Gamma_s \leqslant k \leqslant \Gamma_e \quad (1) \qquad (2)$$

$$t_{(g,l)} \in \{0,1\} \qquad (2)$$

$$t_{(g,m)} \geqslant t_{(g,n)}, \qquad \forall m \leqslant n \quad (3)$$

The constraint (1) in Eq. (2) means that the sum of the scheduled packets $s_{(\hat{g},l)}$ from Γ_s to any instant k should be smaller than

the sum of available upload bandwidth from Γ_s to this instant k. $\overline{U}_g$ is equal to the available upload bandwidth in generation g, and it is equal to the $\overline{U}$ because the average upload bandwidth is considered to remain stable during the transmission period. The constraint (2) means that each class (g,l) can only be transmitted or not transmitted. The constraint (3) means that the lower layer needs to be chosen before the higher layer is chosen.

4.2. Single generation scheduling

The class subscription algorithm selects an optimal layer for each receiver to subscribe for each single generation. To provide a faster and easier solution to class subscription, we propose a single generation scheduling algorithm based on the Eq. (2). This problem finds the layer subscription policy based on the bandwidth information in a single generation.

$$t^*_{(g,l)} = \arg\max_{t_{(g,l)}} \sum_{l=0}^{L} Q_{(g,l)} t_{(g,l)}$$

$$\text{subject to } \sum_{l=0}^{L} t_{(g,l)} s_{(\hat{g},l)} \leqslant \overline{U}_g \quad (1) \tag{3}$$

$$t_{(g,l)} \in \{0,1\} \quad (2)$$

$$t_{(g,m)} \geqslant t_{(g,n)}, \quad \forall m \leqslant n \quad (3)$$

In the Eq. (3), the classes (g,l) that make $t_{(g,l)} = 1$ are chosen as the subscribed layers. To solve the single generation scheduling problem, the $\sum_{k=0}^{l} s_{(\hat{g},k)}$ is compared with $\overline{U_g}$ for any $k \in (0,L)$ in turn. The maximum k is the subscribed layer in generation g.

Algorithm 1. Single Generation Scheduling algorithm

```
for g ← Γ_s to Γ_e do
    for l ← 0 to L do
        if Ū > Σ_{k=0}^{l} s_(ĝ,k) then
            t_g = l ;
        end
    end
end
```

4.3. Multiple generation scheduling

The optimization function of the MGS is a variation of the Multiple-Choice Knapsack Problem [29], which has been proved to be an NP-complete problem [30]. However, the optimal solution can be found in a pseudo polynomial-time through the dynamic programming algorithm [31]. According to the equation proposed in Eq. (2), The constraint (3) in Eq. (2) can be seen as a multiple choice problem. In each generation g, only one quality layer can be targeted. Therefore, the Eq. (2) can be treated as a Multiple-Choice Knapsack Problem and solved using the dynamic programming algorithm as presented in Algorithm.2. The time complexity of this algorithm is $\Gamma_l * \Gamma_l * \overline{U} * L$. Although it is not a polynomial-time solution, it will not increase as the network size increase. Besides, due to the size of the priority region is limited, and the layer of video is limited. The complexity is affordable for each node.

Algorithm 2. Scalable Rate Allocation Algorithm-Find the maximum value

```
Function1 : Optimization function
G = Γ_e − Γ_s; v = Ū * G; R[G+1][v+1] ← 0
for g ← Γ_s to Γ_e do
    for v ← Ū * G to 0 do
        R[g][v] = R[g − 1][min(Ū * (g − 1), v)] ;
        for l ← 0 to L do
            if v > Σ_{k=0}^{l} s_(ĝ,k) then
                R[g][v] = max(R[g][v], Σ_{k=0}^{l} Q_(g,k) + R[g − 1][min((g − 1) *
                Ū, v − Σ_{k=0}^{l} s_(ĝ,k))]
            end
        end
    end
end
return R;
```

In Algorithm 2, we find the maximum PSNR gain we can achieve in Γ. In Algorithm 3, we find the optimal layer subscription $\mathbf{t}$, where $t[g] = -1$ means that this generation is not subscribed, and $t[g] = l$ represents the layer l is subscribed.

Algorithm 3. Find the solution of the optimization function

```
Function2 : Findsolution
remainSpace = Ū * G , t[G] ← 0
for g ← Γ_e to Γ_s do
    t[g] = −1 ;
    for l ← 0 to L do
        if remainSpace >= Σ_{k=0}^{l} s_(ĝ,k) then
            if R[g][remainSpace] − R[g − 1][min(remainSpace −
            Σ_{k=0}^{l} s_(ĝ,k), (g − 1) * Ū)] == Σ_{k=0}^{l} Q_(g,k) then
                t[g] = l ;
                remainSpace = min(remainSpace − Σ_{k=0}^{l} s_(ĝ,k), (g − 1) * Ū)
                ;
            end
        end
    end
end
return t ;
```

After the receiver calculated the optimal class subscription strategy, it sends the strategy to each sender, and then each sender allocates its bandwidth based on the layer subscription policy based on the distributed packet scheduling algorithm proposed in Section 5.

5. Distributed packet scheduling

In this section, the details of the distributed bandwidth-efficient packet scheduling algorithm are clarified. In this algorithm, the

DPS algorithm proposed in our previous work [28] is refined for the scalable data streaming. The scheduling method determines how senders cooperatively contribute their upload bandwidth in transmitting the subscribed layer as shown in Algorithm 4.

Algorithm 4. Distributed Packet Scheduling

$\bar{U} = \sum_{k=0}^{|A_j|} U_{ij}$, $\varsigma_g = \sum_{l=0}^{t_g} s_{(\hat{g},l)}$

for *each sender* $\mathcal{N}_i$ *in* A_j **do**

$d_{ij} = U_{ij}/\bar{U} * \varsigma_g$;

$\bar{U} = \bar{U} - U_{ij}$;

$\varsigma_g = \varsigma_g - d_{ij}$;

end

return **d** ;

In general, the algorithm performs the upload bandwidth allocation for each receiver based on the chronological order that the node U_i becomes the sender of the client node U_j. ς_g is defined as the number of packets that need to be transmitted to the receiver in generation g. It is equal to $\sum_{l=0}^{t_g} s_{(\hat{g},l)}$. $\overline{U}$ is the available upload bandwidth of node N_j with the unit of [packets/generation]. d_{ij} is calculated by the product of the rate of the ith allocated upload bandwidth over all available upload bandwidth. Each sender transmits packets according to the scheduling results d_{ij}.

6. Request model

A request model is proposed to deal with the *unrecoverable* transmission caused by unpredictable network variations. Although the scheduling compensation model considers the peer churn and loss rate in the network, these may vary in a real network. When the receiver node has an unrecovered generation in its priority region, it will periodically broadcast its buffer map as a request signal to all neighboring nodes. One or more request signals from different receivers may arrive at the sender nodes. Then, each sender with spare upload bandwidth capacity calls the "random push algorithm" to push packets to the receivers. When a receiver successfully decodes the corresponding generation, it immediately sends its buffer-map as a stop signal to all neighboring nodes. This mechanism aims at improving the recoverability of the streaming in the system. To keep the efficiencies of the system, a local information updating procedure is called, and the packet scheduling algorithm is recomputed for this node if more than 10% of packets in the *urgent region* are requested from senders or more than 10% of packets in the *priority region* are *uninformative*.

7. Performance evaluation

7.1. Experiment settings

All experiments are based on the same P2P streaming network to achieve a fair comparison. The network is implemented on the event-based network simulator 2(NS2). All end nodes independently choose its neighbors and then form a randomly connected network. The size of the network is set to be 100 nodes. Each node has 20 nodes as its neighbors. As for the download bandwidth of users, as commonly assumed in other P2P system studies, we simulate a P2P network where only the peer upload bandwidth is the bottleneck. The network loss rate is set to be 0.02 as default.

7.1.1. Testing media

The testing video streaming is Paris sequence encoded with H.264/SVC using Medium Grain Scalability (MGS). In our scenario, we have a single source node, streaming a video encoded with H.264/SVC using Medium Grain Scalability (MGS) at CIF resolution. The source node is the only node that is not consuming the video. We make use of the Joint Scalable Video Model (JSVM) reference software to encode scalable video. Three priority classes are selected, exploiting quality scalability, where the base layer has QP equal to 30 and the enhancement layer has QP equal to 22 and MGS vector partitions equal to 6 and 10, for enhancement layer 1 and 2, respectively. We use the Paris sequence with CIF spatial resolution ($352 * 288$) and 30 frames per second. The compressed video streams account for an average bit rates and PSNR of:

	Bitrate [KBps]	PSNR [dB]
Base layer	41.6881	35.6259
Enhancement layer 1	78.7615	37.3439
Enhancement layer 2	113.7042	40.12

We use an I-frame period of 32 frames (corresponding to 1.067 s of video) and a GOP size of 8 frames (equal to 0.2667 s of video). The video GOPs have PSNR and sizes shown in Fig. 4. Fig. 5 also shows the periodic presence of GOPs which are bigger in size, although yielding to approximately the same PSNR of the other GOPs at the same quality layer. The large GOPs correspond to those containing the I-slice, and the periodicity is given by the ratio between I-frame period and GOP length. It should be noted that this is allowed in SVC by the presence of a key-picture in every GOP. However, to decode GOPs without I-frame at a higher quality than the neighboring GOPs in the same I-frame period, one should make sure that the prediction configuration and the coding loops allow that.

Generation size c is 8 frames, which correspond to a group of pictures covering 0.26 s of video. Therefore, κ is 0.26 in this experiment. The number of packets in each generation varies from 19 to 64 packets per generation based on the encoding result of the video, and the average number of packets in each generation is 32 packets in average. The size of priority period is $8c$. It means that the *playback deadline* at each peer is 2.13 s and video packets need to arrive at each node in 2.13 s to not expire. The maximum upload bandwidth of streaming source is set to satisfy 15% end users. The actual size of the video block is 1024 Bytes. Coefficients of network-encoded packets are stored in the packet header and transmitted with the video packets as well, whose size depends on the number of encoding packets in the generation. In NS2 simulation, the packet header (e.g. the address of destination peer, and the sequence number of video) is 150 Bytes, and the average size of network coding coefficients is about 32 Byte per packet (1 Byte for the coding coefficient of each video packet in the generation). Basically, in such experimental settings, the average upload bandwidth need to be at least 1.18 times ς to achieve full rate streaming.

7.1.2. Performance comparison

In this section, our proposed single generation optimization scheduling (SGS) and multiple generation scheduling (MGS) is compared with the hierarchical network coding approach (HNC) [12], and a ADS algorithm [26]. In HNC, each node randomly pushes its encoded packets to its neighboring nodes according to the buffer-map of its neighboring nodes. Different from the R2 for non-scalable video streaming, in scalable video streaming, the enhancement layers are pushed to each receiver node only when the base layers are successfully decoded by this receiver node.

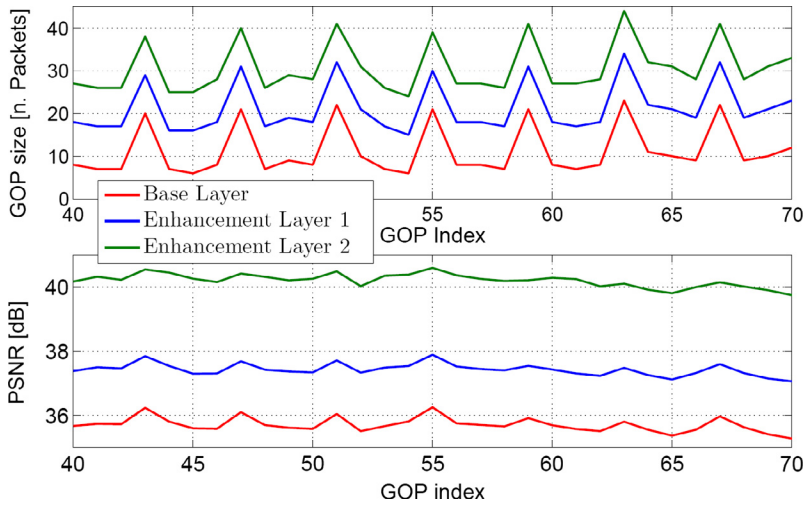

Fig. 4. Paris CIF video, H.264/SVC with Medium Grain Scalability: Size of GOPs (top) and PSNR (bottom).

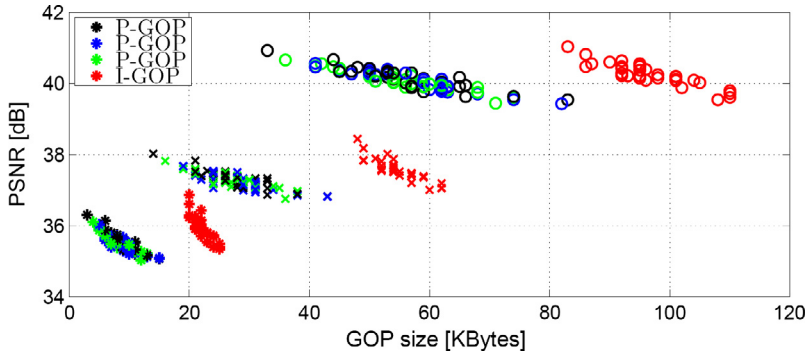

Fig. 5. Paris CIF video, H.264/SVC with Medium Grain Scalability: PSNR vs Size of GOPs for different classes of GOPs.

The buffer-map which represents the data availability of its neighboring peers updates every 100 ms. Furthermore, based on the HNC algorithm, to improve the informative transmission ratio over the network, two more implementations are added. First, a 15% buffer threshold for each generation is used before senders push packets. Second, the number of received packets and the number of sent packets from the receiver node and the sender node are compared to avoid useless transmission. In ADS, each sender independently chooses the optimal transmission policy from candidate transmission policies. This choice is based on the cost of each candidate policy. This transmission policy determines which generation and which receiver the packet is addressed to. The cost is defined as the product of the number of packets needed to recover the generation g and the corresponding reminding time of this generation g before the playback deadline. All candidate policies are sorted in an increasing order according to the defined cost. The optimal policy is selected from the top 30 policies with uniform probability. For each receiver, the lower layer is pushed first.

In the following experiments, we only consider a static P2P network, where peers do not churn during simulation. Our evaluations focus on the following metrics: (1) Average video quality comparison: The average received peak signal to noise ratio (PSNR) at each client node. (2) Delivery Ratio: The average fraction of the average received informative packet that could arrive at the receiver node before the playback point at each client node. (3) The uninformative ratio: The ratio of the number of uninformative video packets to the number of total received video packets. (4) Network size: The impact of network size on the average video quality. (5) Network loss: The impact of network loss on the average video quality.

7.2. Streaming performance comparison

To evaluate the QoS of the proposed streaming system, the performance of the average video quality is studied first. It is the most visual indicator to the quality of service. As depicted in Fig. 6, the video quality of these four methods are compared when the peer upload rate ranges from 0.1 to 1.4 times the full rate video. In all, the results in Fig. 6 show that the proposed multi-generation scheduling (MGS) scheme achieves a remarkable improvement in video quality over the whole range of bandwidth values. The SGS has a poor performance when the upload bandwidth is less than $0.26S$ and has a better performance when the upload bandwidth is better than $0.26S$ compared with other schemes, because the MGS and the SGS transfer packets based on the DPS algorithm. In this way, the delay of the signaling message won't bring great impact to the system performance. In comparison, the ADS and the HNC algorithm transmit packets based on the signaling message. In our experiment, the time delay between the time that the receiver receives enough packets to the time that senders receive this stop message is 100 ms. Therefore, after the receiver successfully decodes a layer in a generation, senders may still push *uninformative* packets to the receiver due to the information updating delay. These uninformative packets are one of the main reasons which caused bandwidth inefficiency. As the MGS algorithm is the

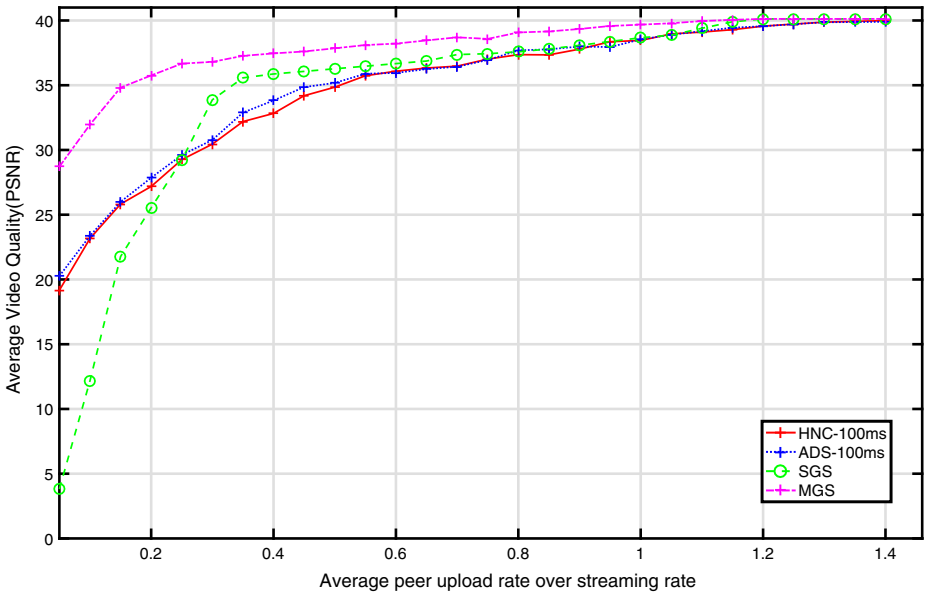

Fig. 6. Performance comparison of the average video quality among four schemes.

optimal solution to the Eq. (2), it can provide the best bandwidth allocation solution to optimize the overall PSNR for each receiver node. Therefore, it can outperform the SGS, HNC and the ADS algorithm. Furthermore, when the average upload bandwidth is from 0.1*S* to 0.25*S*, the HNC and the ADS outperform the SGS algorithm because the SGS will transmit nothing according to the Eq. (3). It means the algorithm will consider many generations can be successfully decoded based on available better chances to transmit decodable generations and layers. When the upload bandwidth is larger than the 0.25*S*, the SGS can have a better performance than the HNC and ADS.

Next in Fig. 7, we study the performance of packet delivery ratio of each method when the peer upload rate ranges from 0.1 to 1.4 times the full rate video. Although the QoS is analyzed in the video quality analysis, then the delivery ratio of each layer can give a more clear understanding of the bandwidth allocation among each layer. Firstly, among all these scheduling algorithms, the MGS has the best delivery ratio performance for each layer in a given upload bandwidth. It achieves full delivery of the base layer when the peer upload bandwidth is 0.2*S*. In comparison, the SGS, the HNC and the ADS achieve the full delivery of the base layer when the peer upload bandwidth is 0.35, 0.8 and 1.0 respectively. Therefore, the SGS achieves faster full delivery than the ADS and the HNC algorithm. However, the ADS performs better than the SGS algorithm when the upload bandwidth is from 0*S* to 0.2*S*. When applying the SGS scheduling algorithm, the system considers the available bandwidth cannot afford the base layer according to Eq. (3). It, therefore, determines to transmit nothing to avoid uninformative transmission. In contrast, the ADS algorithm constantly pushes packets based on the rank of each transmission policy. Therefore, it may have some probability that some generation can be successfully decoded, and thereby bringing better delivery ratio than the SGS. When the upload bandwidth increases, the SGS will think the system can afford the corresponding layers and start the

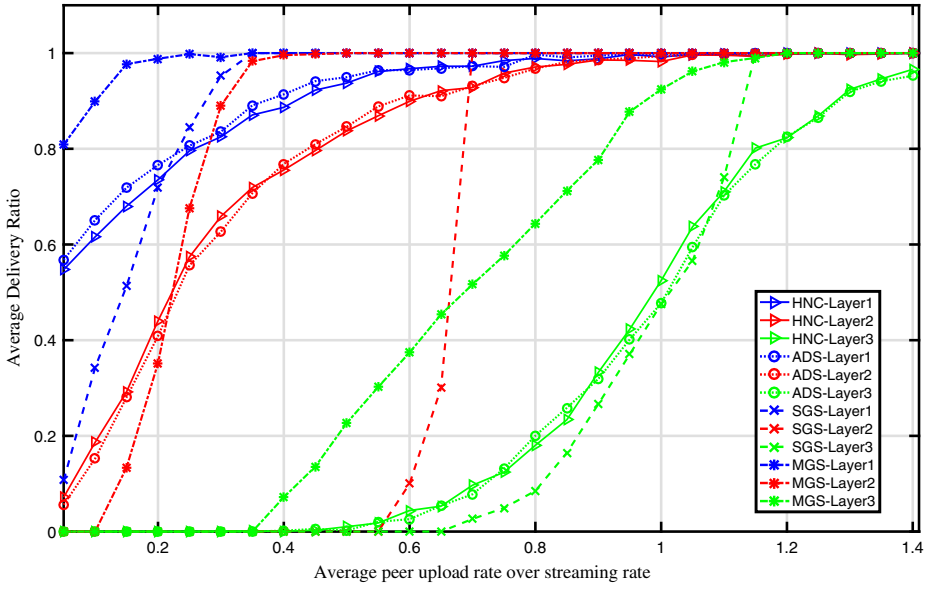

Fig. 7. Performance comparison of the average delivery ratio among four schemes.

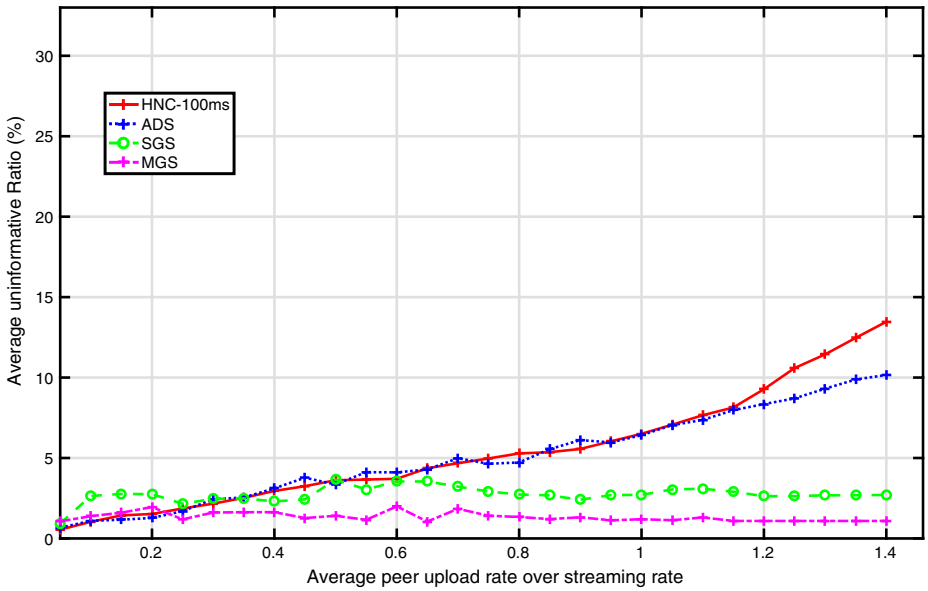

Fig. 8. Performance comparison of the average uninformative packet ratio among four schemes.

transmission. As the DPS algorithm can better cooperate senders and avoid resources inefficiency, it outperforms the ADS when the upload bandwidth is larger than 0.2*S*. It is worth noting that the MGS does not have this issue because it uses the multi-generation scheduling algorithm, which means that the upload bandwidth in a period would be properly allocated. Some GOP may get totally skipped to ensure the overall maximized video quality when the upload bandwidth is low.

We further study the influence of the upload bandwidth on the performance of average uninformative packet ratio. The uninformative ratio is defined as the *uninformative* video packet rate over all transmitted video packet rate. It can reflect the impact of the signaling message delay to the bandwidth redundancy. The results in Fig. 8 shows that the uninformative ratio is 1–2%, 1–3%, 1–10% and 1–13.5% for MGS, SGS, ADS and HNC respectively. In all, using the buffer-map update to control the date delivery would generate more uninformative information compared with the DPS algorithm, especially when the upload bandwidth increase. Furthermore, this problem is more serious in the scalable video streaming system because the number of video packets in each layer The MGS and SGS use the DPS transmission policy, therefore it can reduce these uninformative transmissions. Although the DPS will not generate any uninformative packet transmission due to the late signaling message, it still transmit a certain amount of uninformative packets $s_{(\hat{g},l)} - s_{(g,l)}$ on purpose to avoid the unsuccessful transmission caused by network loss and inherent uninformative transmission caused by network coding.

In the next experiment, we compare in Fig. 9 the average received video quality of the proposed algorithm versus the number of peers to observe the scalability of the packet scheduling algorithm. The perceived video quality of SGS, MGS, ADS, and HNC almost remain the same when the network size varies. This is because the scalability of the streaming network mainly depends on the network bottleneck. In our comparison, all networks are

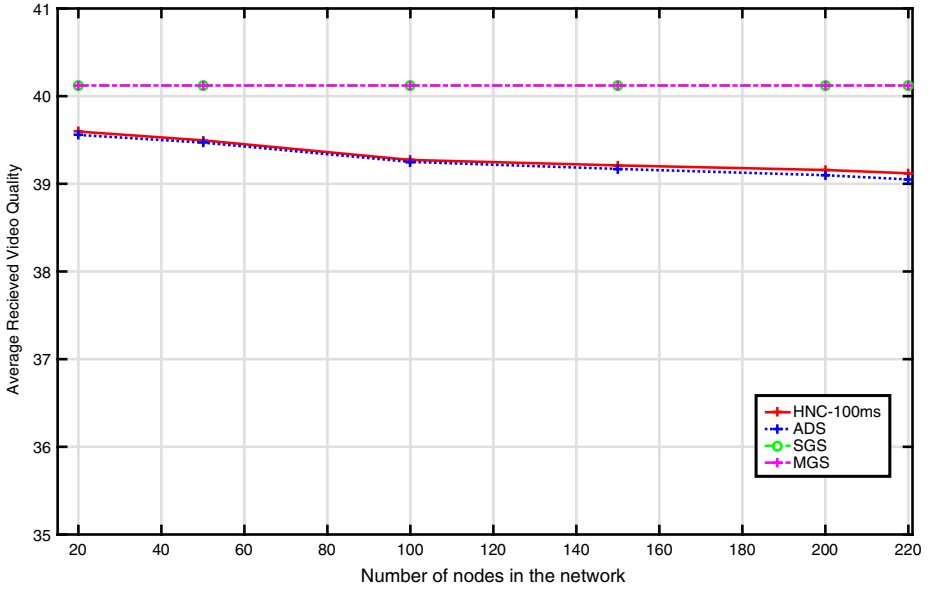

Fig. 9. Performance comparison of the average delivery ratio as a function of network size.

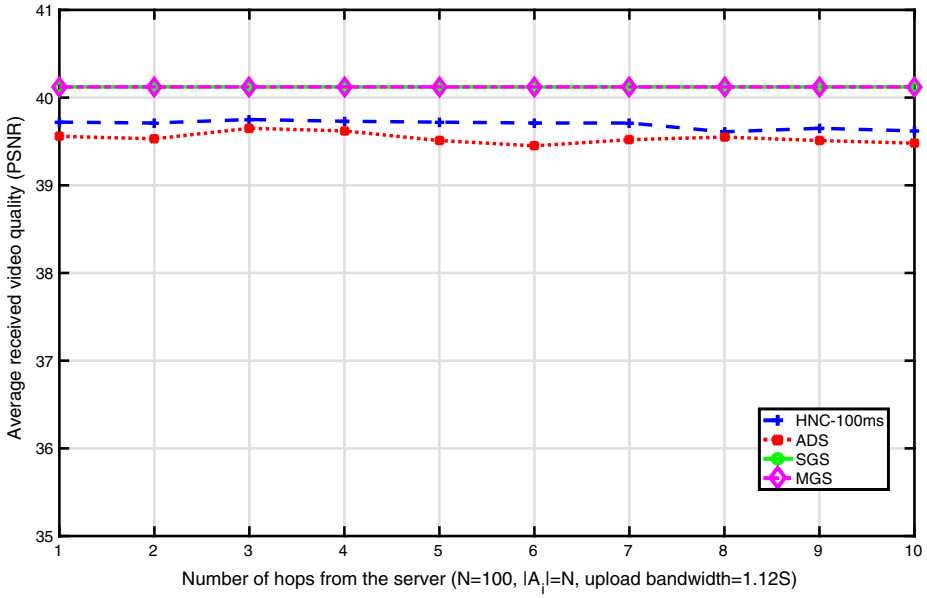

Fig. 10. The impact of distance from server to the average received video quality in PSNR.

formed as random mesh network. Therefore, when there is no network bottleneck in the network, the network performance will remain the same as the network expended. Furthermore, to make sure the system do not have the content bottleneck when performing scalable video streaming, the upload bandwidth of the server is set to be 15% of the overall streaming rate to facilitate fast content distribution so that every node can have new data to distribute among its swarming neighbors.

The performance of the video quality in PSNR versus the number of hops from the server is evaluated in this experiment. More specifically, this experiment demonstrates the scalability of the scheduling algorithm and the fairness among client nodes. The hop is defined as the hop distance from the client node $\mathcal{N}_j$ to the streaming server. As depicted in Fig. 10, the average video quality remains almost the same as the number of hops increases. This shows that all algorithms are not very sensitive to the distance from the server. That is because the HNC, ADS, SGS, and MGS algorithms are built in the mesh network, where the neighbor nodes of each client node are randomly chosen. Therefore, the distance from the streaming server does not have an obvious influence on the delivery ratio.

In Fig. 11 the performance of the delivery ratio is analyzed in a lossy network to evaluate its resistance to network loss. We compare the achieved average received video quality among the four schemes. Through this experiment, we can see that how different algorithms perform against the network loss. ADS and the HNC use the signaling buffer-map to control the content delivery in the lossy network, and the MGS and the SGS algorithms use the scheduling compensation mode (SCM) proposed in [28] to control the content delivery in the lossy network. In all, the scheduling compensation mode is not robust enough when the network loss increases. The SCM use the average loss rate to perform scheduling

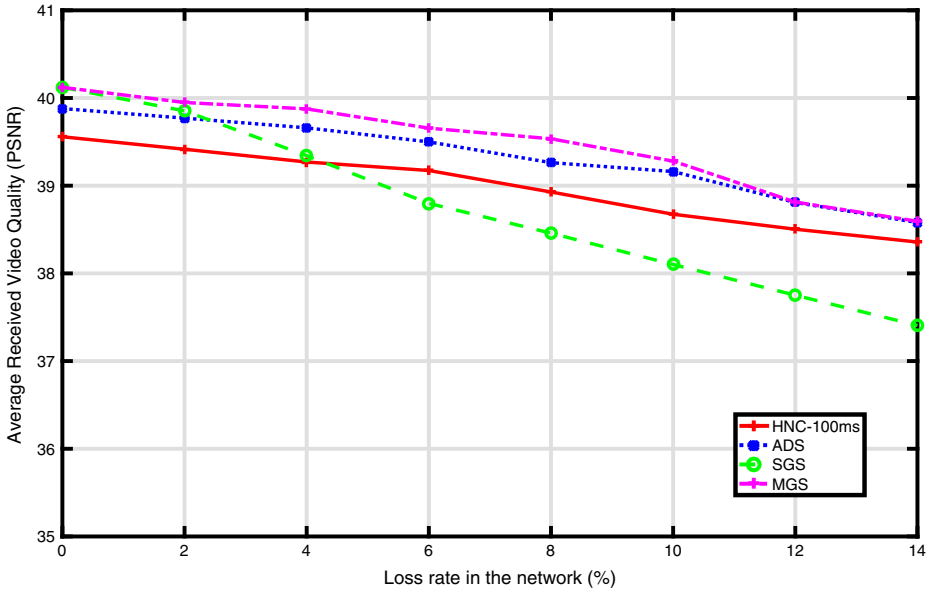

Fig. 11. Performance comparison of the average delivery ratio as a function of network loss rates when $N = 100$ and upload bandwidth $\overline{U} = 1.2S$.

compensation for each generation. However, the scheduling compensation may be inaccurate when the loss rate increase, which leads to redundant transmission, and unsuccessful transmission. Therefore, the HNC and ADS algorithms are more robust than the MGS and SGS when the network loss rate is significant.

8. Conclusions

This paper proposes a novel push-based live scalable streaming system. To accurately perform the packet scheduling and avoid the braking effect during scalable video streaming, we proposed a multiple generation scheduling optimization problem. Based on the optimization problem, two practical generation scheduling algorithms are proposed and implemented. Further, a distributed packet scheduling is proposed to coordinate sender nodes to deliver the selected content. The experimental results prove that the two scheduling algorithms greatly improve the delivery ratio and reduce the uninformative transmission, thereby brings better bandwidth efficiency and quality of service.

References

[1] A.M. Sheikh, A. Fiandrotti, E. Magli, Distributed scheduling for scalable P2P video streaming with network coding, in: INFOCOM, 2013 Proceedings IEEE, IEEE, 2013, pp. 11–12.
[2] M. Sanna, E. Izquierdo, Proactive prioritized mixing of scalable video packets in push-based network coding overlays, in: 2013 20th International Packet Video Workshop (PV), IEEE, 2013, pp. 1–7.
[3] T. Ho, M. Médard, R. Koetter, D.R. Karger, M. Effros, J. Shi, B. Leong, A random linear network coding approach to multicast, IEEE Trans. Inform. Theory 52 (10) (2006) 4413–4430.
[4] A. Shokrollahi, Raptor codes, IEEE Trans. Inform. Theory 52 (6) (2006) 2551–2567.
[5] M. Luby, Lt codes, in: Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '02), IEEE, 2002, pp. 271–280.
[6] R. Ahlswede, N. Cai, S.Y. Li, R.W. Yeung, Network information flow, IEEE Trans. Inform. Theory 46 (4) (2000) 1204–1216.
[7] P.A. Chou, Y. Wu, K. Jain, Practical Network Coding.
[8] E. Magli, M. Wang, P. Frossard, A. Markopoulou, Network Coding Meets Multimedia: A Review. Available from: arXiv:1211.4206.
[9] Z. Liu, C. Wu, B. Li, S. Zhao, UUSee: large-scale operational on-demand streaming with random network coding, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[10] L. Yu, L. Gao, J. Zhao, X. Wang, SonicVoD: a VCR-supported P2P-VoD system with network coding, IEEE Trans. Consum. Electron. 55 (2) (2009) 576–582.
[11] D. Sejdinović, D. Vukobratović, A. Doufexi, R.J. Piechocki, et al., Expanding window fountain codes for unequal error protection, IEEE Trans. Commun. 57 (9) (2009) 2510–2516.
[12] K. Nguyen, T. Nguyen, S.-C. Cheung, Peer-to-peer streaming with hierarchical network coding, in: 2007 IEEE International Conference on Multimedia and Expo, IEEE, 2007, pp. 396–399.
[13] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, F. Zhang, Lion: layered overlay multicast with network coding, IEEE Trans. Multimedia 8 (5) (2006) 1021–1032.
[14] Bittorent, [Online] <http://www.bittorrent.com/>.
[15] M. Zhang, L. Sun, X. Xi, S. Yang, iGridMedia: providing delay-guaranteed peer-to-peer live streaming service on internet, in: IEEE Global Telecommunications Conference (GLOBECOM), 2008, pp. 1–5.
[16] Z. Liu, Y. Shen, K.W. Ross, S.S. Panwar, Y. Wang, LayerP2P: using layered video chunks in P2P live streaming, IEEE Trans. Multimedia 11 (7) (2009) 1340–1352.
[17] Z. Shen, J. Luo, R. Zimmermann, A.V. Vasilakos, Peer-to-peer media streaming: insights and new developments, Proc. IEEE 99 (12) (2011) 2089–2109.
[18] N. Magharei, R. Rejaie, Y. Guo, Mesh or multiple-tree: a comparative study of live P2P streaming approaches, in: INFOCOM 2007, 26th IEEE International Conference on Computer Communications, IEEE, 2007, pp. 1424–1432.
[19] X. Zhang, H. Hassanein, A survey of peer-to-peer live video streaming schemes – an algorithmic perspective, Comput. Netw. 56 (15) (2012) 3548–3579.
[20] B. Li, Z. Wang, J. Liu, W. Zhu, Two decades of internet video streaming: a retrospective view, ACM Trans. Multimedia Comput. Commun. Appl. 9 (1s) (2013) 33.
[21] X. Zhang, J. Liu, B. Li, Y.S. Yum, CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2005, pp. 2102–2111.
[22] M. Wang, B. Li, R2: random push with random network coding in live peer-to-peer streaming, IEEE J. Sel. Areas Commun. 25 (9) (2007) 1655–1666.
[23] S. Huang, M. Sanna, E. Izquierdo, P. Hao, Optimized scalable video transmission over P2P network with hierarchical network coding, in: 2014 IEEE International Conference on Image Processing (ICIP), IEEE, 2014, pp. 3993–3997.
[24] A.T. Nguyen, B. Li, F. Eliassen, Chameleon: adaptive peer-to-peer streaming with network coding, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[25] N. Thomos, J. Chakareski, P. Frossard, Prioritized distributed video delivery with randomized network coding, IEEE Trans. Multimedia 13 (4) (2011) 776–787.
[26] A.M. Sheikh, A. Fiandrotti, E. Magli, Distributed scheduling for low-delay and loss-resilient media streaming with network coding, IEEE Trans. Multimedia 16 (8) (2014) 2294–2306.
[27] N. Thomos, E. Kurdoglu, P. Frossard, M. Van der Schaar, Adaptive prioritized random linear coding and scheduling for layered data delivery from multiple servers, IEEE Trans. Multimedia 17 (6) (2015) 893–906.
[28] S. Huang, E. Izquierdo, P. Hao, Bandwidth-efficient packet scheduling for live streaming with network coding, IEEE Trans. Multimedia (4) (2016) 752–763.
[29] D. Pisinger, A minimal algorithm for the multiple-choice knapsack problem, Euro. J. Oper. Res. 83 (2) (1995) 394–410.
[30] H. Kellerer, U. Pferschy, D. Pisinger, Introduction to NP-Completeness of Knapsack Problems, Springer, 2004.
[31] D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, Dynamic Programming and Optimal Control, vol. 1, Athena Scientific Belmont, MA, 1995.