Bandwidth-Efficient Packet Scheduling for Live Streaming With Network Coding

Huang, S; Izquierdo, E; Hao, P

Bandwidth-efficient Packet Scheduling for Live Streaming with Network Coding

Shenglan Huang, *Student Member, IEEE,* Ebroul Izquierdo, *Senior Member, IEEE,* Pengwei Hao, *Member, IEEE*

***Abstract*—Network coding (NC) brings substantial improvements in terms of throughput and delay in collaborative media streaming applications. A key aspect of NC-driven live peer-to-peer streaming is the packet scheduling policy. Indeed, lack of synchronization among peers usually results in significantly redundant packet transmission, which in turn leads to severe bandwidth inefficiencies. In this paper we address the problem of finding a suitable asynchronous packet scheduling policy that greatly helps to overcome this critical redundant transmission problem. We propose a bandwidth cost minimization technique under a full video packet recovery constraint. In order to add a scalability and improved performance, we also further derive a distributed packet scheduling algorithm. Both implementation and analytical considerations of the proposed approaches are described in this paper. Experimental results confirm that the proposed algorithms deliver higher bandwidth efficiency with reduced redundancy and communication overhead rate, and consequently, better quality-of-service in terms of improved video quality and delivery ratio.**

***Index Terms*—Live broadcasting, Peer-to-Peer TV systems, Network coding, Distributed scheduling**

I. INTRODUCTION

Over the last decade, live video streaming has been considered as an Internet killer. This prevalent view is due to network users spending significant time in online communication sharing media and watching videos streamed over the Internet. Indeed, multimedia communications are pervasive and a substantial part of modern life. Many studies have been conducted in the field of media transmission for a multitude of applications including: remote immersive and interpersonal communication, e-Learning, video conferences, and interactive entertainment. In the field of media transmission, many previous studies have been conducted in multi-source streaming trees construction and scalable video streaming [**?**], [**?**]. Comparatively little attention has been given to improving bandwidth efficiencies through suitable packet scheduling schemes. However, improving bandwidth efficiency is necessary as more effective bandwidth usage may lead to a better quality of service in media streaming applications. This paper therefore focuses on improving bandwidth efficiency in cooperative live streaming networks.

Network coding was first introduced in Information Theory by Ahlswede et al. [**?**], who proved that the maximum capacity of a network can be achieved by transmitting mixed data at intermediate nodes. Based on the theoretical model, Chou et al.[**?**] proposed a practical network coding scheme for media streaming. In their work, the network topology and the coding function at each node are unknown by other nodes. Each node performs a linear combination of its incoming packets using random coefficients over the Galois field to generate the outgoing packet. These random coefficients are stored in the header of the outgoing packet. When a client node receives enough linear independent packets, it can recover the original packet by solving a conventional system of linear equations. Their work proves that practical network coding can provide significant gains in throughput and delay. Many large-scale applications of utilizing network coding in the field of multimedia streaming [**?**], [**?**], [**?**] have also demonstrated the associated in reducing communications delays and facilitating the cooperation among nodes.

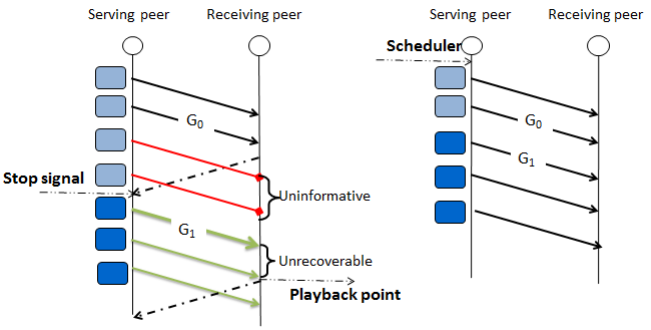
Peer-to-peer (P2P) networks have been used extensively in multimedia live streaming as an effective transmission platform. In P2P live streaming, peers collaboratively organize themselves into an overlay and contribute their upload capacities to others. P2P networks are classified into two types: (i) the tree-based, and (ii) the mesh-based P2P network [**?**], [**?**]. The tree-based network [**?**], [**?**], [**?**], [**?**], [**?**] organizes peers into one or more multicast trees. Nodes in the upper layer are the parents of nodes in the lower layer. The original media content is decomposed into sub-streams and pushed from parent nodes to their child nodes without explicit requests from child nodes. Tree-based networks minimize the delay of distributing media content, however, they are too complex to maintain in the face of peer churns and bandwidth variations. In contrast, *mesh-based* networks use gossip-like protocols to discover content availability among peers. In mesh-based networks, nodes form a mesh by holding a list of neighbouring peers. The original streaming content is treated as a series of segments. A *buffer-map* which represents the data availability of each segment is exchanged among peers to *pull* or *push* segments among neighbouring nodes. The mesh-based P2P network with the *pull* model is also called the *mesh-pull* network [**?**], [**?**]. The advantage of the mesh-based network is its resistance to peer churn, which enables fast recovery from delivery failures.

Although most commercial P2P systems are based on the *mesh-pull* model like[**?**] and [**?**], they still suffer from the problem of long playback delay caused by the complex communication mechanism. Therefore many studies have been conducted in applying network coding to the mesh-based P2P streaming system to solve this problem. By taking advantage of network coding, *mesh-push* schemes are proposed. The *mesh-push* network provides low playback delay, and high resistance to peer churn and network loss[**?**], [**?**]. For instance, in R^2 [**?**], random encoded segments are pushed to random receivers until the arrival of a stop message. In that paper, the authors show that mesh-push with network coding brings

a superior performance when compared to traditional mesh-pull schemes. More specifically, better delivery ratio and lower transmission delay are reported when the available upload bandwidth exceeds the bandwidth demand. This is shown to hold true even when the available upload bandwidth excess is large. These advantages of *mesh-push* schemes make it very competitive in the field of media streaming. Moreover, to take further advantage of the *mesh-push* schemes, many optimizations have also been made to meet practical needs. For instance, many authors have combined scalable video coding and network coding [**?**], [**?**], [**?**], [**?**], others have focused on reducing the transmission and decoding delay for live streaming applications[**?**], and others have improved performance in error correction and throughput in the lossy network [**?**], [**?**].

Despite substantial progress in the field, little attention has been given to the bandwidth efficiency in the *mesh-push* schemes. Improving bandwidth efficiency is critical for video streaming because efficient bandwidth usage means that more useful packets arrive at receivers. This in turn results in a better quality of service in any multimedia streaming. A problem with current *mesh-push* schemes is that some transmitted packets are redundant in improving received media quality. Let us consider a transmission process in conventional *mesh-push* P2P networks. Senders actively push encoded packets until the arrival of a stop message from the receiver. The choice of packets may be unintelligent in some cases, due to the delayed stop message or actual network conditions. For instance, some transmitted packets are received after the current content generation has already been decoded. We call such packets **uninformative**. Furthermore in some cases, available packets are insufficient to decode the original content within the given time window. Let us term these packets **unrecoverable**. These two types of packets lead to bandwidth inefficiencies. In conventional push-based schemes, it is impossible to avoid the transmission of *uninformative* and *unrecoverable* packets because fully intelligent scheduling requires enormous amounts of overheads for communication which cannot be compensated by the gains achieved. Thus it is critical to find a trade-off between the improved transmission efficiency and the information overheads needed to achieve it. With the goal in mind, we therefore propose a scheduling compensation model (SCM), an adaptive push algorithm (APA), a centralized packet scheduler (CPS), and a distributed packet scheduler (DPS). Firstly, the SCM and APA calculate the number of packets that each receiver could receive from its neighbouring nodes after taking the dynamic network conditions into account. The two algorithms work together to reduce *unrecoverable* transmissions. Secondly, the CPS and the DPS construct the centralized and the distributed multi-sender cooperation models separately. They accurately determine the number of packets that should be sent from each sender to each receiver. The objective of the multi-sender cooperation model is to reduce *uninformative* transmissions caused by the redundant packet transmission. In this way, these redundant transmissions can be reduced, and network bandwidth resources can be better utilized.

The remainder of this paper is organized as follows. In Section II related works on the packet scheduling problem are introduced. The overall system model is presented in Section III. In Section IV, the centralized and the distributed packet scheduling algorithm are presented. In Section V selected results of extensive simulation experiments are presented. The paper is concluded in Section VI.



a)Random push with random network coding

b)Scheduled push with random network coding

Fig. 1. Comparison of the transmission mechanism between the RND scheme and the proposed scheme

II. RELATED WORK

Many studies have been carried out combining scalable video coding (SVC) with network coding (NC) to provide a better quality of experience for people using manifold end-user devices. For instance, the authors in [**?**] propose a prioritized video streaming system that solves the bandwidth allocation problem among different classes of media priority. The authors in [**?**] combine SVC with NC to adapt to peer heterogeneity. In [**?**] the authors calculate the average upload bandwidth of senders to choose a suitable media layer to stream. In [**?**] inter-layer network coding is performed on different layers of SVC to gain greater flexibility in optimizing the data flow.

In recent years, research has also been conducted using NC to achieve faster content delivery. For instance, the authors in [**?**] combine network coding with tree-based P2P network. Like traditional tree-based models, their work treats the whole stream as a sequence of sub-streams. The idea behind their scheduling model is to consider the sub-stream scheduling problem among parent nodes as a max-weighted bipartite matching problem (MWBM), by using the delay cost as the weight, to find the suitable match between the sender i and the substream j. Consequently, the overall transmission delay can be minimized. However, maintaining such multicast trees is difficult in a real-world network. In [**?**], the authors construct a random multicast tree (RMT) to provide a short start-up delay and faster data distribution. Nevertheless, the precondition of RMT is rather restricted and unrealistic. Each jointed peer is required to upload a same amount of data as it received. In [**?**], the authors construct a cost function for each possible transmitting policy, and the sender then chooses the transmission policy based on the calculated cost. Nevertheless, this still cannot avoid the bandwidth inefficiencies caused by these *unrecoverable* packets.

The main difference between the traditional push mechanisms and our scheduled push mechanism is outlined in Fig.1. This illustration shows an extremely simplified transmission process in which a sender node sends encoded media packets to a receiver node. Fig. 1 (a) shows a random push with random network coding algorithm (RND) [**?**], which is used to represent the existing push mechanisms. Fig. 1 (b) is the proposed packet scheduler. The transmissions of these two algorithms are compared to understand their difference. In the illustration, G_0 and G_1 are different generations of a video stream. To decode the original content, the receiver needs to receive two packets of G_0 and three packets of G_1 within the given time window. In the RND algorithm shown in Fig. 1 (a), the sender keeps pushing network-encoded packets of G_0 until the arrival of a stop message from the receiver. However, due to the message updating interval, the stop message from the receiver is delayed. Two *uninformative* packets are sent to the receiver. The sender then starts transmitting network-encoded packets of G_1. At this time, the sender does not have enough upload bandwidth to transmit 3 packets of G_1 within the given time window. For this reason, the received packets are not enough to be decoded. Therefore, the received packets are discarded, wasting network resources. In contrast, in our packet scheduling algorithm shown in Fig. 1 (b), the sender transmits packets according to the results of pre-scheduling. Firstly, the SCM calculates the number of actually needed packets for each generation, and the APA accurately finds the generations that need to be skipped. Next, the CPS and the DPS construct linear programming functions to get scheduling results before the actual media transmission. The scheduling results specify how multiple senders cooperatively transmit packets to receivers. Finally, each sender follows the scheduling results to deliver packets. These *uninformative* transmissions caused by the asynchronous communication can be minimized in this way. Furthermore if some transmitted packets are lost or some senders churn during transmission, the transmission system can achieve fast recovery by pulling packets from other neighbouring senders, since all transmitted packets are network-encoded.

The key to the transmission model is the accuracy of the underpinning pre-calculation. To achieve an accurate pre-calculation of generation skips and scheduling compensations, the actual network conditions such as loss rate, peer churn rate, inherent *uninformative* packet rate, and the actual upload bandwidth are taken into consideration. Subsequently, to achieve an accurately cooperative packet scheduling among multiple senders, two approaches are proposed: a centralized scheduling algorithm (CPS) and a distributed scheduling algorithm (DPS). In general, both scheduling algorithms achieve accurate calculations by estimating a decoding status and allocating the suitable upload bandwidth to receivers. The difference between them is the amount of required information. CPS calculates the packet scheduling strategy based on the global information while DPS calculates the scheduling strategy based on the local information of neighbouring peers. Furthermore, the CPS algorithm formulates the allocation problem as a global integer linear programming (ILP) problem. In contrast, the DPS transforms the global ILP problem into a local ILP problem.

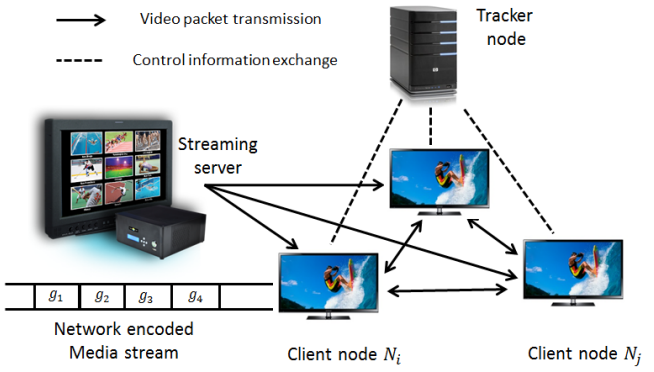


Fig. 2. Illustration of the overall network model of a mesh-based P2P network

It in turn adds a critical scalability property to the packet scheduling model. The experimental results confirm that both algorithms generate less communication traffic, *uninformative* and *unrecoverable* packets compared to conventional random-push schemes.

III. SYSTEM MODEL

The overall system includes a network model, a network-encoded media streaming model, and a packet scheduling model. This section gives the definitions of the network model and the media streaming model.

A. Network Model

The overall network model is depicted in Fig.2. Three types of peers are defined in this model: the tracker node, the streaming server, and the client node. In the system, the tracker node does not transmit any coded video packet, but only serves the purpose of enabling peers to find each other and exchange control messages. Each peer contacts the tracker node to join the network. Some control information is exchanged during the procedure, i.e. the upload bandwidth of neighbor nodes and size of video packets. The peer churn rate in the network consists of both the peer arrival and the departure rate. To simplify the model, the peer churn rate ι is set to be constant. When a sender node leaves the network, another available sender is designated to receivers by the tracker node. The streaming server transmits network-encoded video packets to client nodes. The overlay of network nodes is represented as a graph $(\mathcal{N}, \varepsilon)$ composed for nodes $\mathcal{N} = \{\mathcal{N}_0, ..., \mathcal{N}_N\}$ and the edge ε. In these nodes, $\mathcal{N}_0$ represents the tracker node and $\mathcal{N}_1$ represents the streaming server. The rest nodes from $\mathcal{N}_2$ to $\mathcal{N}_N$ are client nodes. For any nodes in the actual video streaming network, $U = \{U_1, ...U_N\}$ is indicated as the vector of upload bandwidth of streaming nodes. $\bar{U} = \sum_{i=1}^{N} U_i/N$ is defined as the average upload bandwidth of the streaming network. To simplify the discussion, we also define the κ-quantized upload bandwidth as $\tilde{U}_i = U_i * \kappa/V$, where κ is defined as the playback duration of a generation, V is the video packet size. The unit of $\tilde{U}_i$ is the number of packets /κseconds. The average value of the κ-quantized upload bandwidth is also defined for simplicity as $\hat{U} = \sum_{i=1}^{N} \tilde{U}_i/N$. A client node becomes a sender node when it holds α linear independent

segments ($0 \leq \alpha \leq 1$). It can ensure the content availability at the senders. For any node $\mathcal{N}_j$, $A_j \subset \mathcal{N}$ is defined as the neighborhood of $\mathcal{N}_j$. It is the set of sender nodes that are connected to $\mathcal{N}_j$. The child node $\mathcal{N}_j$ reports the network change to the tracker node when it experiences bandwidth variations, such as the departure of a parent, or the bandwidth change.

B. Network-encoded Media Streaming

The media stream that distributed to the network nodes is modeled as a single dimensional array of generations. A generation is usually made of 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 g is subdivided into P_g blocks of symbols, and the size of each video packet is V bytes. Generations with identical temporal index g have the same playback duration κ. The average streaming rate is termed as S. Instead of transmitting raw video packets, nodes transmit linear combinations of its received packets to other nodes. A new outgoing packet is generated by performing random network coding in a single generation [**?**].

A new transmission region is proposed to achieve a scheduled and guaranteed transmission. The transmission region can be divided into an *urgent region* and a *priority region* as shown in Fig.3. The *priority region* is the scheduled transmission region. In the *priority region*, the transmissions of generations are scheduled firstly according to the CPS and the DPS. Senders then follow the scheduling results to transmit packets. In the *urgent region*, *unrecoverable* generations are requested by receiver nodes from its all neighbouring senders $\forall \mathcal{N}_i \in A_j$ according to a request model. A *priority region* Γ is defined as a moving time window next to the *urgent region*. The size of the priority region is Γ_l. The start point and the end point of the *priority region* are Γ_s and Γ_e respectively. An *urgent region* is defined as a moving time window next to the playback point. The start point and the end point of the *urgent region* are ω_s and ω_e. The urgent region and the priority region move as the playback point moves. The example in Fig.3 illustrates that node $\mathcal{N}_j$ is playing the video in generation 3. Its urgent region is from generation 4 to 5 ($\omega_s = 4$ and $\omega_e = 5$). Its priority region is from generation 6 to 11 ($\Gamma_s = 6$, $\Gamma_l = 6$, and $\Gamma_e = 11$). To successfully decode the original media, the client node needs to receive $[P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}]$ informative packets in $g \in (4, 11)$ respectively. For the sake of more clarity, all notations used in the model are summarized in Table I.

C. Scheduling Compensation Model (SCM)

To reduce the *unrecoverable* generations caused by the unsuccessful packet delivery, we develop a scheduling compensation model. Its main function is to calculate the number of packets $\tilde{P}_g$ that need to be sent from $\mathcal{N}_i \in A_j$ to $\mathcal{N}_j$ such that P_g independent packets can successfully arrive at $\mathcal{N}_j$. In a dynamic network, the number of sent packets $\tilde{P}_g$ needs to be larger than P_g so that $\mathcal{N}_j$ can successfully receive P_g packets to decode the original generation. $\tilde{P}_g - P_g$ packets are used

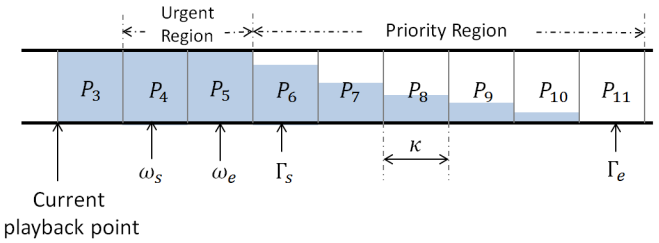


Fig. 3. Sample buffer status of a client node $\mathcal{N}_j$. The current playback point is g=5, the priority region $\Gamma = [\Gamma_6, \Gamma_7, ..., \Gamma_{11}]$.

TABLE I
NOTATION USED IN THE SYSTEM MODEL

$\mathcal{N}_i$	Network node i, $i \in [0, N]$
A_j	Neighborhood of the receiver node $\mathcal{N}_j$
g	Temporal index (generation) in the media, $g \in [1, G]$
κ	Playback duration of each generation
V	Size of video packets
U_i	Upload bandwidth of streaming node $\mathcal{N}_i$, $i \in [1, N]$ [kByte/s]
$\tilde{U}_i$	κ-quantized upload bandwidth of $\mathcal{N}_i, i \in [1, N]$ [packets/κsec]
$\tilde{U}_{ij}$	κ-quantized upload bandwidth allocated from $\mathcal{N}_i$ to $\mathcal{N}_j$ [packets/κsec]
$\bar{U}$	Average upload bandwidth of the streaming network [Kbyte/s]
$\hat{U}$	Average κ-quantized upload bandwidth [packets/κsec]
S	Average streaming rate
α	Independent threshold for a receiver node to become a sender node
Γ	Priority region, $\Gamma = [\Gamma_s, ..., \Gamma_e]$
ω	Urgent region, $\omega = [\omega_s, ...\omega_e]$
μ	Independent transmission rate
θ	Loss rate
ι	Peer churn rate
ρ	Inherent linear independent probability
P_g	Number of real video packets in each generation g
$\tilde{P}_g$	Expected number of packets that $\mathcal{N}_i \in A_j$ must send to $\mathcal{N}_j$ to recover g
$\hat{P}_g$	Number of actually scheduled packets in generation g
$\bar{P}$	Average value of $\hat{P}_g$
H_{ijg}	Number of scheduled packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ in generation g
S_{ijg}	Number of sent packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ in generation g

to compensate the unsuccessful packet delivery caused by the dependent transmission, packet loss, and peer churn.

To find the suitable $\tilde{P}_g$, we use the loss rate θ, peer churn rate ι and the inherent independent probability ρ to estimate the successful transmission rate. We also define the successful transmission rate μ ($0 \leq \mu \leq 1$) as the probability that a linear independent packet is successfully received by a node. Therefore, accounting for the unsuccessful transmission rate, the expected number of scheduled packets that $\mathcal{N}_j$ must receive from $\mathcal{N}_i \in A_j$ to recover the generation g can be written as:

$$\tilde{P}_g = P_g / \mu \tag{1}$$

The successful transmission rate μ is related to the loss rate θ, the peer churn rate ι and inherent linear dependent rate ρ of random network coding. The inherent linear dependent rate ρ is defined as the dependent transmission probability caused by the inherent property of random network coding. Randomly

chosen coefficients may be the same as the coefficients in previously sent packets. According to [**?**], the lower bound of the inherent linear independent probability is $\rho \geq (1 - 2^{-q})$. The lower bound of the successful transmission rate can therefore be simply given as:

$$\begin{aligned} \mu &= (1-\theta)(1-\iota)\rho \\ &\geq (1-\theta)(1-\iota)(1-2^{-q}) \end{aligned} \tag{2}$$

According to Eq.2 and Eq.1, $\mathcal{N}_i \in A_j$ must send to $\mathcal{N}_j$ at least $\tilde{P}_g$ packets to compensate the unsuccessful transmissions so that receivers can successfully decode P_g packets in generation g.

D. Adaptive Push Algorithm (APA)

The APA is proposed to reduce *unrecoverable* transmissions caused by the insufficient upload bandwidth. Reducing *unrecoverable* transmissions can lead to a better use of resources, thereby resulting in better bandwidth efficiencies. The APA reduces *unrecoverable* transmissions by determining the number of actually scheduled packets $\hat{P}_g$ from $\mathcal{N}_i \in A_j$ to $\mathcal{N}_j$ considering the actually available upload bandwidth. These *unrecoverable* packets are generated because the receiver $\mathcal{N}_j$ fails to receive P_g informative packets from senders $\mathcal{N}_i \in A_j$ before the playback deadline, due to limited upload bandwidth restricting the packets sent from the neighbouring nodes. To avoid *unrecoverable* transmissions, the APA assesses if the available κ-quantized upload bandwidth $\hat{U}_g^{Cum}$ can afford the expected transmission $\tilde{P}_g$. When $\hat{U}_g^{Cum}$ can afford the expected transmission $\tilde{P}_g$, generation g is transmitted ($\hat{P}_g = \tilde{P}_g$). Otherwise, the generation is skipped ($\hat{P}_g = 0$). When a generation g is skipped, the average upload bandwidth of g can be used to deliver generations from $g+1$ to Γ_e. The APA can be described by the following pseudo-code in the Algorithm 1:

Algorithm 1: Adaptive Push Algorithm

Input: $\tilde{\boldsymbol{P}}$,$\hat{U}$
Output: $\hat{\boldsymbol{P}}$
for g *from* Γ_s **to** Γ_e **do**
  $\hat{U}_g^{Cum} = \hat{U}_{g-1}^{Cum} + \hat{U}$
  if $\hat{U}_g^{Cum} >= \tilde{P}_g$ **then**
    $\hat{P}_g = \tilde{P}_g$
  else
    $\hat{P}_g = 0$
  end
  $\hat{U}_g^{Cum} = \hat{U}_g^{Cum} - \hat{P}_g$
  Store element $\hat{P}_g$ into vector $\hat{\boldsymbol{P}}$
end
return $\hat{\boldsymbol{P}}$;

The APA is performed by the *track server* before the transmission. For every generation, the tracker node compares the expected number of packets $\tilde{P}_g$ with the available κ-quantized upload bandwidth $\hat{U}_g^{Cum}$. When $\hat{U}_g^{Cum} \geq \tilde{P}_g$, $\hat{P}_g = \tilde{P}_g$, otherwise, $\hat{P}_g = 0$. In this way only recoverable generations are transmitted, and the bandwidth is better utilized. $\hat{U}_g$ is the average κ-quantized upload bandwidth of the streaming network in generation g. It is equal to the $\hat{U}$ in our system. The unit of $\hat{U}_g^{Cum}$, $\hat{P}_g$ and $\hat{P}_g$ are packets/κseconds. To inform all client nodes of the number of actually scheduled packets $\hat{P}_g$, the tracker node needs to send a message to each client node $\mathcal{N}_i$ every generation. The size of a communication message is 2 bytes. The communication overhead of APA is $2N$ bytes/κseconds, which is $2N/\kappa$ bytes/s. $\bar{P}$ is the average value of $\hat{P}_g$. If the streaming rate is considered as a relatively stable rate, P_g packets can successfully arrive at each receiver when $\hat{U} \geq \bar{P}$.

IV. Optimized Packet Scheduler

This section proposes two packet schedulers to determine how multiple senders cooperatively contribute their upload bandwidth to different receivers. The schedulers can reduce the *uninformative* transmission caused by asynchronous communications, thereby leading to better transmission efficiencies. The process of the packet scheduling can be summarized as follows: the packet schedulers calculate the number of packets that each sender needs to send to its receiver, and then each sender follows the scheduling results to transmit packets individually. The objective of a streaming network is that senders cooperatively deliver all recoverable generations to all receivers and avoid *uninformative* transmissions at the same time. This requires each sender to accurately determine the number of packets to send to each receiver at each generation. For this purpose, two packet schedulers are proposed. Firstly, a centralized packet scheduler (CPS) is proposed to improve the global bandwidth efficiency. The CPS formulates the global packet scheduling problem as a global integer linear programming problem. A distributed packet scheduler is then derived from the CPS by finding solutions to an approximative optimization objective. Both CPS and DPS accurately schedule packet transmissions, and senders can cooperatively contribute their upload bandwidth to receivers to achieve recoverable and informative transmissions.

A. Centralized Packet Scheduler (CPS)

The CPS is a global packet scheduler, designed to find the global multi-sender cooperation model to organize the upload bandwidth of all senders to their neighbouring receivers. Such a multi-sender cooperation problem is modelled as a redundant transmission minimization function among all senders under a constraint of full recovery. *Uninformative* transmission is caused by the superfluous transmission after the corresponding generation is already successfully transmitted. The streaming network can successfully transmit all recoverable generations with the CPS, and simultaneously minimize *uninformative* transmissions. Any receiver node $\mathcal{N}_j$ needs to receive $\hat{P}_g$ packets to recover generation g. and any other packets are *uninformative* for the client nodes. The objective function of the CPS is to therefore minimize the number of transmitted packets, and the constraints of the CPS are to ensure the full transmission of $\hat{P}_g$.

The number of packets that should be sent from each sender $\mathcal{N}_i$ to each receiver node $\mathcal{N}_j$ in generation $g \in \Gamma$ is denoted as a non-negative integer H_{ijg}. A positive constant C is defined as the transmission cost (the consumed bandwidth of a single packet). For each generation $g \in \Gamma$, the scheduling problem is formulated as a cost minimization problem with some given restrictions in Eq.3, and the minimization function is solved as an Integer Linear Programming (ILP) problem by the Simplex method.

$$
\begin{aligned}
\boldsymbol{H} = \arg\min_{H_{ijg}} C \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} H_{ijg} \\
\text{subject to} \quad \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} H_{ijg} \leq \sum_{g=\Gamma_s}^{\Gamma_e} \tilde{U}_{ig} \quad \forall i \ \text{(a)} \\
\sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \quad \forall j, g \ \text{(b)} \\
H_{ijg} \leq \alpha \hat{P}_g \quad \forall i \in (2, N),\ j,\ g \ \text{(c)} \\
H_{ijg} = 0 \quad \forall \mathcal{N}_i \notin A_j,\ j,\ g \quad \text{(d)}
\end{aligned}
\tag{3}
$$

The optimization objective is to find the minimum bandwidth cost to achieve the full delivery of $\hat{P}_g$. The constraint (a) means that the number of sent packets of each sender node $\mathcal{N}_i$ should be smaller than its available κ quantized upload bandwidth $\tilde{U}_{ig}$ in the priority period, where $\tilde{U}_{ig}$ is equal to $\tilde{U}_i$. The constraint (b) means that senders need to cooperatively send at least $\hat{P}_g$ packets so that P_g packets can arrive at the receiver node. The constraint (c) means that the number of scheduled packets from $\mathcal{N}_i$ to $\mathcal{N}_j$ needs to be smaller than the number of its *linear independent* packets $\alpha\hat{P}_g$. This constraint guarantees that enough contents are available at each client node. The constraint (d) guarantees that only nodes $\mathcal{N}_i \in A_j$ can send packets to its neighbouring nodes $\mathcal{N}_j$. The optimization function achieves its optimized solution at $\sum_{i=1}^{N} H_{ijg} = P_g$.

This algorithm hypothesizes that a central coordinator which knows all upload bandwidth $\tilde{U}_i$ of each node $\mathcal{N}_i$ exits in the network. It could be the tracker node in our system because it holds all information of client peers in the network. In the global scheduling algorithm, the streaming server is scheduled like other sender peers so that a precise multi-sender cooperation model can be built for each receiver. After the CPS calculates the number of packets that should be sent from each sender $\mathcal{N}_i$ to each receiver $\mathcal{N}_j$, the tracker node sends a control message to each sender $\mathcal{N}_i$ The scheduling results are then stored in the local record of each sender node. The number of sent packets S_{ijg} and H_{ijg} are compared when every transmission opportunity arises at the sender node $\mathcal{N}_i$. If S_{ijg} is smaller than H_{ijg}, a network encoded packet in generation g is sent to node $\mathcal{N}_j$ and the value of S_{ijg} increases. Generations close to Γ_s are those that are initially pushed.

B. Distributed Packet Scheduler (DPS)

The DPS is proposed to increase the scalability of the system. The DPS is a distributed solution to the centralized optimization function. It solves how multiple senders contribute their bandwidth to each receiver based only on the local bandwidth information. As mentioned in section IV-A, the CPS algorithm globally reduces *uninformative* transmission, thereby improving bandwidth efficiencies. However, in a large-scale P2P network, the complexity of the global optimization would be extremely high. Therefore a distributed bandwidth-efficient packet scheduling algorithm is proposed in this section. In the DPS, we find the symbolic solution to the packet scheduling problem by providing some constraints to an approximate optimization objective of the CPS. The main steps of this algorithm include: 1) Each sender allocates its upload bandwidth to each receiver by the handshake procedure; 2) Each sender uses a local weighted quadratic equation to find the most suitable allocation for each receiver in each generation.

1) Handshake Procedure: In the handshake procedure, each sender node $\mathcal{N}_i$ allocates its overall κ-quantized upload bandwidth $\tilde{U}_i$ to its receiver node $\mathcal{N}_j$, denoted as $\tilde{U}_{ij}$. The unit of the allocated upload bandwidth U_{ij} is packets/κseconds. When the receiver node $\mathcal{N}_j$ joins the network, it initially contacts the tracker node to obtain a list of free sender peers $\mathcal{N}_i$ to form its neighbouring nodes A_j. When the tracker picks the set of sender nodes, it ensures that $\sum_{\mathcal{N}_i \in A_j} \tilde{U}_i \geq \bar{P}$. The tracker node then calculates the amount of upload bandwidth $\tilde{U}_{ij}$ that each sender $\mathcal{N}_i$ should contribute to support the delivery to this receiver node $\mathcal{N}_j$ according to the Eq.4.

$$
\tilde{U}_{ij} = \min\left(\frac{\tilde{U}_i}{\sum_{\mathcal{N}_i \in A_j} \tilde{U}_i} * \bar{P}, \alpha * \bar{P}\right) \tag{4}
$$

If $\sum_{\mathcal{N}_i \in A_j} \tilde{U}_{ij} < \bar{P}$, more senders are allocated to this receiver and the Eq.4 is recomputed until $\sum_{\mathcal{N}_i \in A_j} \tilde{U}_{ij} \geq \bar{P}$. Note $\tilde{U}_{ij} = 0$ for any $\mathcal{N}_i \notin A_j$. The scheduled upload bandwidth $\tilde{U}_{ij}$s for any $\mathcal{N}_i \in A_j$ are stored into a vector following its identification order and sent to each sender node $\mathcal{N}_i \in A_j$. At the same time, the tracker node calculates the new available upload bandwidth for each sender $\mathcal{N}_i \in A_j$ by $\tilde{U}_i' = \tilde{U}_i - \tilde{U}_{ij}$. At the end of the handshake procedure, each sender $\mathcal{N}_i$ will get a list of cooperative senders $\mathcal{N}_i \in A_j$, and their corresponding allocated bandwidth $\tilde{U}_{ij}$ for this receiver $\mathcal{N}_j$.

2) Real-time Distributed Scheduler: The real-time distributed scheduler pre-calculates the number of scheduled packets H_{ijg} from node $\mathcal{N}_i$ to $\mathcal{N}_j$ in any generation g. The cooperative transmission model in the local scheduling algorithm can be summarized as a multi-sender and single receiver relationship. A receiver node $\mathcal{N}_j$ has a set of sender nodes $\mathcal{N}_i$, where $\mathcal{N}_i \in A_j$. The optimization problem is formed as a bandwidth cost optimization function under the full recovery constraint in Eq. 5. This optimization function calculates the number of packets H_{ijg} that should be sent from $\mathcal{N}_i$ to $\mathcal{N}_j$ so that $\hat{P}_g$ packets can be successfully decoded at the receiver node.

$$\boldsymbol{H} = \arg\min_{H_{ijg}} \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}}$$
$$\text{subject to} \sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \quad \forall j, g \tag{5}$$

The above optimization function finds the suitable allocation H_{ijg} subject to a constraint that the number of scheduled packets must be larger than $\hat{P}_g$. This constraint ensures that this receiver node receives enough packets to recover the original content in generation g. As the transmission allocation to each node $\mathcal{N}_j$ in each generation is independent, Eq.5 can be transformed to Eq.6.

$$\boldsymbol{H} = \arg\min_{H_{ijg}} \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}}$$
$$\text{subject to} \sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \quad \forall j, g \tag{6}$$

The above optimization problem can be solved by finding the solution $\boldsymbol{H}$ that minimizes the expected Lagrangian in Eq. 7 since $\sum_{i=0}^{N} H_{ijg}$ approaches to $\hat{P}_g$.

$$J(\boldsymbol{D}) = \sum_{i=1}^{N} \frac{(\tilde{U}_{ij} - H_{ijg})^2}{\tilde{U}_{ij}} + \lambda(\hat{P}_g - \sum_{i=1}^{N} H_{ijg}) \forall j, g \tag{7}$$

By solving the above Lagrangian function, the symbolic solution H_{ijg} from sender $\mathcal{N}_i$ to any receiver node for any generation g can be expressed as followed in Eq. 8:

$$H_{ijg} = round(\frac{\tilde{U}_{ij}}{\sum_{k=i}^{I} \tilde{U}_{kj}} P_r) \quad (a)$$
$$P_r = \hat{P}_g - \sum_{k=1}^{i-1} H_{kjg} \qquad (b) \tag{8}$$

In Eq. 8 (a), H_{ijg} is calculated by the product of the bandwidth ratio and the remaining number of packets P_r. The ratio can be expressed as the ith allocated upload bandwidth over all allocated upload bandwidth from sender node $\tilde{U}_{ij}$ to the last sender node $\tilde{U}_{Ij}$. Eq.8 (b) means that the remaining number of packets P_r can be represented by the difference between the scheduled result $\hat{P}_g$ and the number of scheduled results for sender node $\tilde{U}_0$ to $\tilde{U}_{i-1}$.

The optimization objective in the DPS is the approximation of the optimization objective in the CPS algorithm. According to the optimization function, it is clear to see that the proposed optimization objective in DPS is the square of the H_{ijg}. Moreover, the solution to Eq.5 also satisfies all constraints in Eq.3. The details of proof can be found in Appendix A.

In general, the distributed packet scheduling algorithm uses the handshake procedure to do the bandwidth pre-allocation for each sender. The multilevel linear programming can therefore be written as a single layer linear programming. By solving the single layer linear programming, the symbolic solution to the ILP problem can be found, thereby reducing the computational complexity. Furthermore, the solution to the quadratic equation is weighted, which means that sender nodes with small $\tilde{U}_{ij}$ have less variation. This helps to reduce the scheduling error. After each sender gets the scheduling results H_{ijg}, it calculates the number of pending transmitting packets by subtracting S_{ijg} from H_{ijg}. If H_{ijg} is larger than S_{ijg}, an encoded packet is sent to the receiver. Generations closed to Γ_s are pushed initially.

In all, the distributed scheduling algorithm gives a symbolic solution to the local optimization function. More, the solution of the distributed algorithm also satisfies the constraints of CPS algorithm in the Eq. 3. Both CPS and DPS reduce the redundant transmission by global or local packet scheduling. The network resources can be better utilized, and the bandwidth efficiencies can be improved.

C. 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 urgent region, it will periodically broadcast its buffer map as a request signal to all neighbouring 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 decode the corresponding generation, it immediately sends its buffer-map as a stop signal to all neighbouring 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*.

V. PERFORMANCE EVALUATION

In this section, we report experimental results with our packet schedulers using streaming a network-encoded media sequence over a mesh-push based P2P network. To evaluate the performance of the proposed scheduling algorithms, we compare our proposed packet schedulers with three existing scheduling algorithms with four different sets of considerations. Firstly, the quality-of-service (QoS) at the client nodes is measured for video quality and delivery ratio, with different upload bandwidths. Secondly, the bandwidth efficiencies in the network are measured in terms of *uninformative ratio*, communication overhead, and informative packet rate. Thirdly, the performance in a lossy network is evaluated, and finally, the system scalability is evaluated. We present the simulation environment and metrics in Section V-A and the streaming performance comparison in Section V-B.

A. Experimental Settings and Metrics

We evaluate our packet schedulers over a mesh-push based P2P network. The P2P network is implemented on an event-based network simulator NS2. All end nodes independently choose its neighbors and then form a randomly connected network. The size of the default test network is set to be 100 nodes. The neighborhood size A_j is set to be 20 nodes. The default upload bandwidth of the streaming server U_1 is set satisfy 15% end-users in the network. The setting of the server upload bandwidth is similar to the settings in [**?**] to achieve a fair comparison. In the presented three first experiments, the upload bandwidth of each peer is set to be a constant value ranging from $0.8S$ to $1.45S$. Here we aim at evaluating the QoS and bandwidth efficiencies of the system. S is the encoded video rate. These initial experiments proved that when the peer upload bandwidth is $1.2S$, the delivery ratio can reach 99.9%. Therefore, in the remaining experiments we set the peer upload bandwidth to be $1.2S$. This enabled us to observe the impact of other parameters in the performance of the proposed approach. The default loss rate and the peer churn rate are set to be 0.05. The default value of α is set to be 15%.

The test video stream is the Paris sequence encoded with H.264/AVC. The average bit rate of stream S is 116 Kbyte/s. The average video quality of the media is 40.12dB. The size of a generation κ is 8 frames, which corresponds to a group of pictures covering 0.26 seconds of video. On average, there are 32 packets in each generation. The size of the priority region Γ_l is 8κ. The size of the urgent region is 4κ. This indicates that the *playback deadline* at each peer is 3.12 seconds. All packets must arrive at client nodes in 3.12 second to be successfully played. The actual size of a video block is 1024 Bytes. Packets in each generation are encoded using network coding over the Galois Field **GF**(2^8). Coefficients of network-encoded packets are stored in the packet header and transmitted with each video packet as well, whose size depends on the number of encoding packets in the generation. In the 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 Bytes per packet (1 Byte for the coding coefficient in GF(2^8) per packet). Theoretically, in such settings, the average streaming rate should be at least equal to $1.18S$ to achieve the full rate streaming.

We compare our proposed packet schedulers with the following approaches:

- Advanced random push with random network coding with 50ms buffer-map updating interval (ARND-50ms): The ARND is an improved version of RND similar to [**?**]. In RND, each sender randomly pushes its encoded packets to its neighbors according to the buffer-map of its neighbors. In ARND, senders tend to choose those generations that need more packets to be decoded before the playback deadline. The *buffer-map* which represents the data availability of its neighbor peers updates every 50ms.
- Advanced random push with random network coding with 200ms buffer-map update interval (ARND-200ms): The ARND algorithm with 200ms buffer-map updating interval.
- An asynchronous distributed scheduler (ADS) similar to [**?**]: 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.

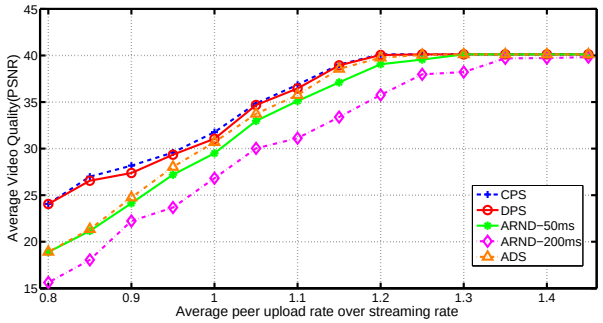


Fig. 4. Performance comparison of the average video quality with five schemes

We use the following performance metrics to evaluate the QoS and bandwidth efficiencies: (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 *recoverable* packets that could arrive at the receiver node over the actual streaming packet rate before the playback point at each client node; (3) the *uninformative* packet ratio: the received *uninformative* video packet rate over all received video packet rate; (4) communication overhead ratio: the communication overhead rate over the streaming rate. Communication overhead includes buffer-map exchange messages, scheduling messages and packet request messages; (5) informative packet rate: the number of received informative packets per second.

B. 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. Generally, the video quality increases when the peer upload bandwidth increases from $0.8S$ to $1.45S$. The results in Fig. 4 show that the CPS and the DPS perform better than the other schemes over the whole range of the upload bandwidth, especially when the upload bandwidth is too low for full-rate video delivery. When the peer upload bandwidth is $1.2S$, the CPS and the DPS can achieve 40.12dB, 40.05dB respectively. By contrast, at the same bandwidth, the ADS,

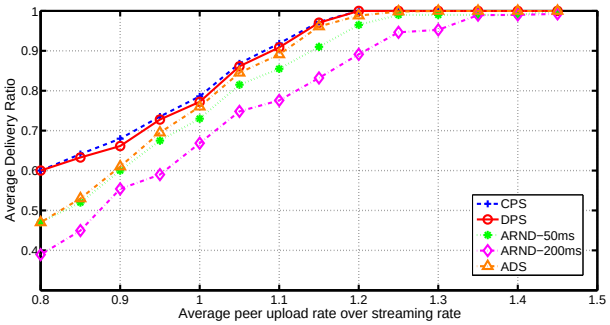


Fig. 5. Performance comparison of the average delivery ratio with five schemes

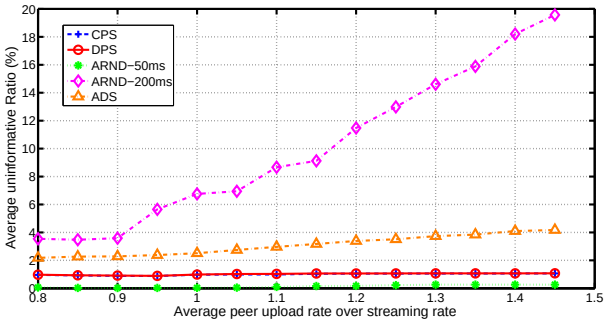


Fig. 6. Performance comparison of the average uninformative packet ratio with five schemes

the ARND-50ms, and the ARND-200ms can only achieve 39.75dB, 39.06dB, and 35.76dB respectively. This is because the communication overhead and *uninformative* transmission waste network bandwidth. When the peer upload bandwidth is $0.8S$, the DPS shows significant improvements over ADS, ARND-50ms, and ARND-200ms algorithms; 5.14dB, 5.16dB, 8.43dB respectively. This is because the ADS and ARND cannot avoid *unrecoverable* transmissions that waste bandwidth resources and lead to poor transmission of other generations. In contrast, the APA effectively skips some generations so that all transmitted packets are recoverable. The DPS achieves slightly lower video quality compared with the CPS algorithm over the whole range of bandwidth values. That is because the CPS is a global optimization algorithm while the DPS is a local optimization algorithm.

In Fig. 5, the performance of packet delivery ratio among these five methods is studied when the peer upload bandwidth ranges from 0.8S to 1.45S. Better delivery ratio will bring smoother playback of videos. In this experiment, when the peer upload bandwidth is $0.8S$, the delivery ratio of CPS, DPS, ADS, ARND-50ms, and ARND-200ms is 61.03%, 60.31%, 47.05%, 47.01% and 38.9% respectively. This demonstrates that our algorithms can improve 13%-22% in delivery ratio compared with those three scheduling algorithms. This improvement is mainly because the proposed APA algorithm can efficiently avoid *uninformative* transmission. Furthermore, the CPS and DPS can achieve about 4% more delivery ratio compared with the ADS algorithm when the upload bandwidth of peers is $1.2S$. This improvement is mainly due to the CPS and the DPS generates less communication overhead and *uninformative* packets. On the whole, this experiment demonstrates that our algorithms can achieve better recoverable packet delivery ratio when the upload bandwidth of peers ranges from $0.8S$ to $1.45S$.

To evaluate the bandwidth efficiencies of the streaming network, we also observe the performance of the *uninformative* packet ratio in Fig. 6. The *uninformative* ratio is an important criteria to the evaluation of bandwidth efficiencies. The experiment shows that the *uninformative* ratio of the CPS, DPS, and ARND-50ms algorithms remains stable under different upload bandwidth, and the *uninformative* ratio of the ARND-200ms increases a lot as the upload bandwidth of peers increases. It can be seen that the ARND-50ms only generates about

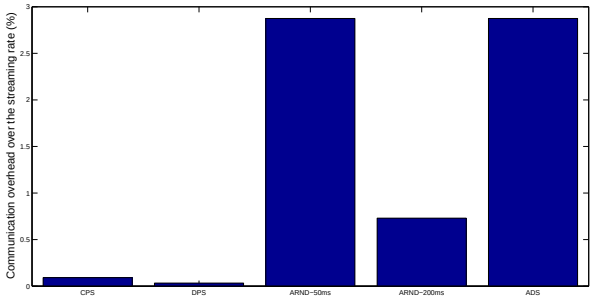


Fig. 7. Performance comparison of the communication overhead over the streaming rate with 5 schemes

0.1% to 0.3% *uninformative* packets, and the CPS and the DPS generate about 1% *uninformative* packets over the whole range of peer upload bandwidth. The ADS generates about 2.2% to 4.1% *uninformative* packets, and the ARND-200ms generates about 3.5% to 19.57% *uninformative* packets. Firstly, it demonstrates that when the buffer-map updating interval of ARND is 50ms, the amount of *uninformative* transmission caused by the lack of synchronization is very small and almost negligible. Secondly, the CPS and the DPS still may generate a small amount of *uninformative* transmission. This is because the request model may bring a small number of *uninformative* transmission. Compared with the ADS and ARND-200ms, the amount of *uninformative* packets is acceptable. Thirdly, the ADS algorithm and the ARND-200ms have a higher *uninformative* ratio. This is because the ADS algorithm has the sub-optimization problem and the long buffer-map updating period of ARND-200ms brings a large number of *uninformative* transmission.

To further evaluate the bandwidth efficiencies of the streaming network, the average communication overhead over the streaming rate is studied in Fig.7. The communication overhead of the ARND and the ADS is the bandwidth resources used for buffer-map updating. The communication overhead of the CPS and the DPS is the bandwidth resources used for packet scheduling and packet request. As depicted in Fig. 7, the CPS and the DPS have a relatively low communication overhead while the ARND-50ms and the ADS have a higher overhead. This is because the ARND-50ms and

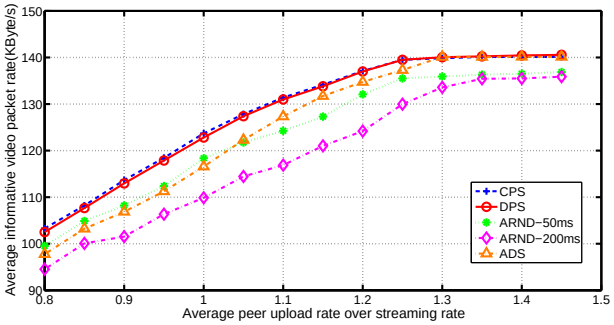


Fig. 8. Performance comparison of the average informative video packet rate with five schemes

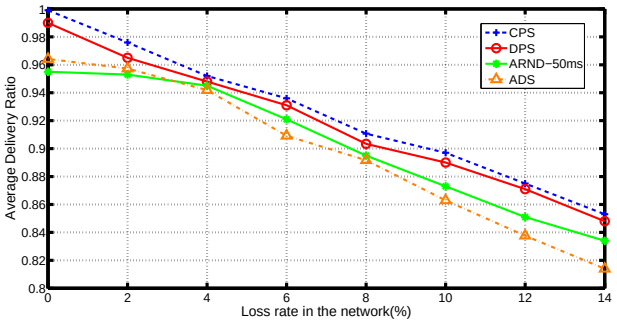


Fig. 10. Performance comparison of the average delivery ratio as a function of network loss rates when $N = 100$ and upload bandwidth $\bar{U} = 1.2S$

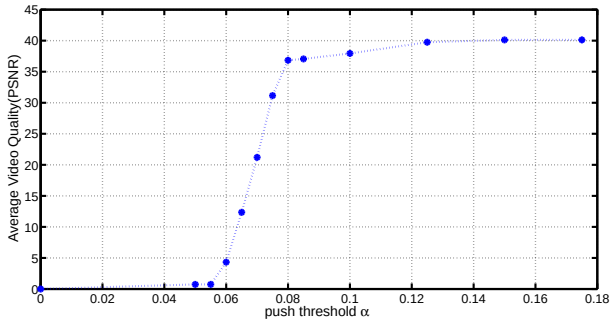


Fig. 9. The average delivery ratio versus different values of α

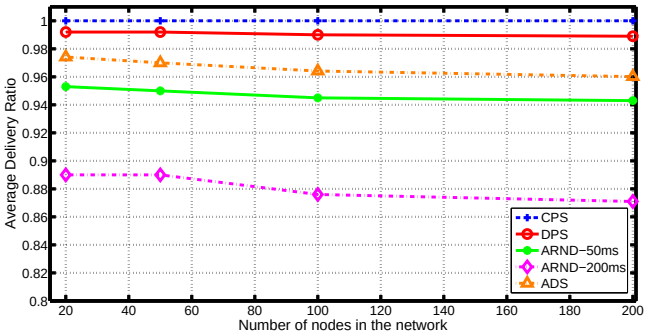


Fig. 11. Performance comparison of the average delivery ratio in networks of different network sizes when the upload bandwidth $\bar{U} = 1.2S$

the ADS algorithm frequently exchange buffer-map among neighbouring nodes. By contrast, the CPS only transmits the scheduling results from the tracker node to each client node and the DPS only exchanges the information among peers during the handshake procedures and when peer nodes experience bandwidth variations. Therefore, compared with the communication overhead of buffer-map updating, the amount of communication overhead of CPS and DPS is relatively small.

We also evaluate the actual average *informative* packet rate in Fig.8 to observe the joint impact of the communication overhead and the *uninformative* transmission when the upload bandwidth of peers increases. It can be seen that the proposed CPS and DPS have a better informative packet rate compared with the other two methods over the whole range of peer upload bandwidth values. Although the CPS and the DPS have about 1% more *uninformative* packet ratio than the ARND-50ms algorithm, the CPS and the DPS get more *informative* packet rate. That is because that the communication overheads of ARND-50ms use more upload bandwidth of peers.

The impact of the threshold α can be seen in Fig.9. This figure shows that a sender node needs to have at least α segments before it becomes a sender of other client nodes. We can see that a lower threshold cannot guarantee that enough linear independent packets are available at the sender nodes, thereby leading to a poor video quality.

In Fig. 10 the performance of the delivery ratio is analyzed in a lossy network to evaluate its resistance to network loss. In this experiment, we increase the loss rate and then observe the change in the delivery ratio. Generally, the delivery ratio decreases when the loss rate increases for all schemes. That is because the useful upload bandwidth decreases when the loss rate increases. The results in Fig. 10 show that the decreasing rate of both CPS and DPS is similar to the decreasing rate of other scheduling algorithms. It proves that the scheduling compensation model can accurately estimate the number of needed packets for client nodes in a lossy network. The loss in the network will not bring extra scheduling errors to CPS and DPS.

In Fig. 11, we evaluate the scalability of our proposed packet schedulers by varying the number of peers from 20 to 200 peers. Fig. 11 shows that CPS and DPS offer steady delivery ratio, whereas the ADS, ARND-50ms, and ARND-200ms have a slightly decreasing delivery ratio as the number of peers increases. It demonstrates that all these algorithms can be applied to a large-scale P2P network. The property of network coding and push-based P2P network make these packet scheduling algorithms very competitive in terms of the scalability.

In the last experiment, the performance of the video quality in PSNR versus the number of hops from the server is evaluated. This experiment can demonstrate 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. 12, the average video quality remains almost the same as the number of hops increases. This shows that all algorithms are not very sensitive

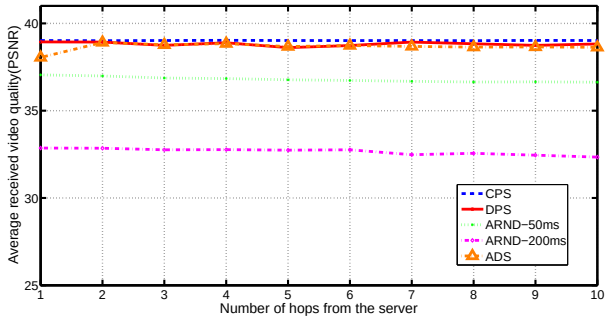


Fig. 12. The impact of distance from server to the average received video quality in PSNR

to the distance from the server. That is because that the CPS, DPS, ADS, and ARND algorithms are built in the *mesh-push* 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.

VI. CONCLUSION

This paper proposes a centralized and a distributed bandwidth-efficient packet scheduling algorithm for multi-media live streaming applications. This packet scheduling algorithm avoids the critical problem of superfluous packet transmission and brings significant improvement in band-width efficiencies. The centralized packet scheduling algo-rithm finds the globally optimal packet scheduling policy that minimizes the overall bandwidth cost. To add a critical scalability property to the proposed approach, we propose a distributed packet scheduling algorithm to find the local optimal solution by constructing a pre-allocation procedure and solving the single layer linear programming problem using the Lagrangian algorithm. Our experiments show that our packet schedulers achieve better video quality and delivery ratio, a lower redundant packet ratio and more informative packets when the peer upload bandwidth varies. Furthermore, the robustness to lossy networks and the network scalability is also proved by different experiments. Our future work will focus on improving the accuracy of scheduling in a more dynamic P2P network.

APPENDIX A
PROOF OF EQUIVALENCE OF OPTIMIZATION FUNCTION

proof: The equivalence of the proposed CPS and the DPS algorithm is proved. This proof includes 2 parts. One is the proof of the equivalence of the two optimization objectives. The other is the proof of the equivalence of constraints.

Part 1: We will prove the optimization objective of the DPS is the approximation of the objective of the CPS algorithm.

$$\begin{aligned} Q(\boldsymbol{H}) &= \sum_{g=\Gamma_s}^{\Gamma_e}\sum_{j=2}^{N}\sum_{i=1}^{N}\frac{(\tilde{U}_{ij}-H_{ijg})^2}{\tilde{U}_{ij}} \\ &= \sum_{g=\Gamma_s}^{\Gamma_e}\sum_{j=2}^{N}\sum_{i=1}^{N}(\tilde{U}_{ij}-2H_{ijg}+\frac{H_{ijg}^2}{\tilde{U}_{ij}}) \end{aligned} \tag{9}$$

As assumed in the constraint of the DPS optimization in Eq.5, we have that

$$\sum_{i=1}^{N} H_{ijg} \geq \hat{P}_g \quad \forall j,g \tag{10}$$

Then we have that

$$Q(\boldsymbol{H}) \leq \sum_{g=\Gamma_s}^{\Gamma_e}\sum_{j=2}^{N}\sum_{i=1}^{N}\tilde{U}_{ij} + \sum_{g=\Gamma_s}^{\Gamma_e}\sum_{j=2}^{N}\sum_{i=1}^{N}\frac{H_{ij}^2}{\tilde{U}_{ij}} - 2\sum_{g=\Gamma_s}^{\Gamma_e}\sum_{j=2}^{N}\hat{P}_g \tag{11}$$

In the expanded function, $\tilde{U}_{ij}$ and $\hat{P}_g$ are known positive constant. The optimization objective of DPS is the p-Norm of H_{ijg} (p=2). The optimization objective of Eq.3 is the 1-norm of H_{ijg}. The property of norm is shown in Eq. 12,

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{2}\|x\|_2 \tag{12}$$

As the vector space is a real finite-dimensional one, all norms are equivalent. So we can get the conclusion that the objective of DPS is the approximation of the objective of CPS. The two optimization function have an approximate solution.

Part 2: We will prove that the solution to the DPS optimiza-tion in Eq.5 satisfies the constraint (a) of CPS alggortihm in Eq.3. Consider

$$H_{ijg} = \frac{\tilde{U}_{ij}}{\sum_{k=1}^{|A_j|}\tilde{U}_{kj}}\hat{P}_g \quad \forall i,j,g \tag{13}$$

As the scheduling algorithm is independent to g, we have that

$$\sum_{g=\Gamma_s}^{\Gamma_e} H_{ijg} = \sum_{g=\Gamma_s}^{\Gamma_e}\frac{\tilde{U}_{ij}}{\sum_{k=1}^{|A_j|}\tilde{U}_{kj}}\hat{P}_g \quad \forall i,j \tag{14}$$

As scheduled in the handshake procedure of the DPS algorithm, we have that

$$\sum_{k=1}^{|A_j|}\tilde{U}_{kj} \geq \bar{P} \quad \forall j \tag{15}$$

so that

$$\sum_{g=\Gamma_s}^{\Gamma_e} H_{ijg} \leq \sum_{g=\Gamma_s}^{\Gamma_e}\frac{\tilde{U}_{ij}}{\bar{P}}\hat{P}_g \quad \forall i,j \tag{16}$$

If we assume that the actual streaming rate p fluctuates in a relatively narrow band around $\bar{P}$, then we have that

$$\sum_{g=\Gamma_s}^{\Gamma_e}\bar{P} = \sum_{g=\Gamma_s}^{\Gamma_e}\hat{P}_g \quad \forall i,j \tag{17}$$

So that

$$\begin{aligned} \sum_{g=\Gamma_s}^{\Gamma_e} H_{ijg} &\leq \sum_{g=\Gamma_s}^{\Gamma_e} \tilde{U}_{ij} \quad \forall i,j \\ \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} H_{ijg} &\leq \sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} \tilde{U}_{ij} \quad \forall i \end{aligned} \tag{18}$$

As supposed in the handshake procedure of DPS algorithm, we have that

$$\sum_{j=2}^{N} U_{ij} \leq \tilde{U}_i \quad \forall i \tag{19}$$

Hence,

$$\sum_{g=\Gamma_s}^{\Gamma_e} \sum_{j=2}^{N} H_{ijg} \leq \sum_{g=\Gamma_s}^{\Gamma_e} \tilde{U}_i \quad \forall i \qquad \square \tag{20}$$