A Push Scheduling Algorithm with Network Coding for Peer-to-Peer Live Streaming

Shenglan Huang, Ebroul Izquierdo, Pengwei Hao

Queen Mary University of London, UK

Abstract

Network coding(NC) appears to bring 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. In previous peer-to-peer network, the *buffer-map* is widely used to pull or push packets from parent nodes to achieve data transmission. However, it often leads to undesirable long playback delay in live streaming applications. In this paper, we propose a new push scheduling policy to address this problem. In the proposed scheme, the packet scheduling is calculated at parent client nodes in advance. Then the parent nodes push the network-encoded packets to the children nodes according to the pre-calculated scheduling algorithm. As a consequence, the proposed packet scheduling algorithm eliminates the processes of *buffer-map* updating and packet requesting, which in turn reduces the number of redundant packets and leads to less traffic of redundant video data over the Internet. The simulation results show that the proposed scheme provides significantly better delivery ratio, less control traffic and fewer redundant packets than conventional push-based schemes.

Keywords: Packets scheduling policy, Network coding, Peer-to-peer streaming

1 Introduction

In recent years, live video streaming application has gained great popularity among users. Peer-to-peer (P2P) network has been widely used as an effective paradigm to delivery multimedia data over the Internet successfully, such as PPLive [1], CoolStreaming/DoNet [16] and iGridMedia [14]. The essential advantage of live peer-to-peer streaming is that the load on dedicated streaming server is significantly mitigated because P2P allows all the upload bandwidth to be fully utilized, which provides end users with a better video watching experience when the bandwidth is limited. Network architecture is important in building P2P networks. According to the network architecture, P2P network can be classified into two types: the tree-based and the swarm-based schemes [15], [5]. Early P2P networks are based on multicast distribution trees, like ZIGZAG [11]. In the tree-based P2P network, the data packets are pushed from upper level to lower level immediately when a new data packets is received. It performs well when participating nodes remain stable. Packets are pushed directly over the whole Internet with less delay and fewer communication packets, but the tree-based network suffers from reconfiguration problem in the dynamic P2P environments. To address this issue, nowadays the swarm-based P2P networks are widely used for live streaming applications like CoolStreaming/DoNet [16] and GridMedia [14]. The swarm-based scheme works in a way that every node connects with other nodes to form the neighbouring relationships. Each peer periodically advertises a bit-map, which represents the availability of video blocks, to its neighbours to obtain the missing blocks.

The two common scheduling strategies in a swarm-based P2P network are the pull-based strategy and the push-based strategy. In some mesh-based models, receivers pull segments from senders according to the *buffer-map*s sent from senders. It is termed as *pull* mode [8], [16], [7], [6]. In some other mesh-based models, senders directly push packets to receivers according to the sent *buffer-map*s from receivers. It is termed as *push* mode.

The *mesh-push* schemes are proposed by taking advantages of network coding. The benefits of applying network coding to mesh schemes are reduced buffering times, and sustained throughput [12], [4]. Recently network coding is regarded as a promising innovation in the communication field and is widely applied for the data transmission. It was firstly proposed in the field of Information Theory by Ahlswede et al. [2] and proved that the maximum capacity in a network can be achieved by transmitting the mixed data at the intermediate nodes, instead of simply relaying the original data to the next node. In peer-to-peer communication system, it has been demonstrated to bring extra advantages by providing easier peer coordination, higher bandwidth utilization, and lower communication delay, which are essential properties to improve the quality of services.

In this paper, we propose a new push scheduling(PS) algorithm that improves the coded peer-to-peer communication by reducing the number of packets exchanged between the senders and the receivers, and by minimizing the redundant packets caused by the end-to-end communication delay. In contrast to previous push-based strategies, the new push scheduling policy does not need peers to ad-

vertise a bit-map to their neighbours to request packets periodically. Instead, every sender executes the PS algorithm to obtain the number of needed packets for each receiver. Less communication traffic and redundant packets are generated over the Internet, compared to other push-based policies.

The remainder of this paper is organized as follows. In Section 2 we introduce other related works on scheduling and explain the scheduling problem in previous network. Then in Section 3 we describe the proposed push scheduling algorithm. In Section 4 we present the results of some simulation experiments, and in Section 5 we draw some conclusions.

2 Related work

In this section we give a brief overview of the main P2P live media streaming systems with their main advantages and disadvantages respectively. The most representative pull-based P2P system in a swarm-based network is CoolStreaming/DoNet[16]. In the network, a video content is divided into several blocks and a *buffer-map* is used to represent the availability of blocks, where a bit 1 or 0 indicates if a block is available or unavailable. Each child node knows the block availability of their parent nodes by requesting a *buffer-map* of them. To pull video packets from parent nodes, each child node makes decisions based on the *buffer-map*, the streaming information, and the local information, and sends a message to parent nodes to request data packets. Its limitation is the long communication delay caused by the pull procedures.

The most effective method to coordinate peers in a push-based scheme is network coding. Wang and Li proposed an algorithm in R2 [13] by combining random push and random network coding together to coordinate packet transmission in the push-based network. In this algorithm, each peer sends network-coded packets to some random picked peers immediately after it receives a new packet. When enough packets are received by a peer, a *buffer-map* which represents the availability of every block is sent to every neighbour node to stop the transmission. R2 shows a dramatic improvement over previous push-based algorithms, but it still faces the bandwidth waste problem, which is caused by the massive traffic packets and the redundant video packets. To further improve the push scheme, a new push scheduling algorithm is developed in this paper to obtain the exact number of packets needed from each sender based on bandwidth estimation, so a large amount of redundant communication packets and useless video packets can be avoided. A transmission delay comparison of these algorithms is illustrated in the Figure 1.

Packet scheduling design is crucial to P2P live streaming. Chan et al. proposed a network-encoded packets scheduling algorithm to achieve fast recovery in substreaming push algorithm.[3] Different from their work, our push scheduling based on packets instead of sub-

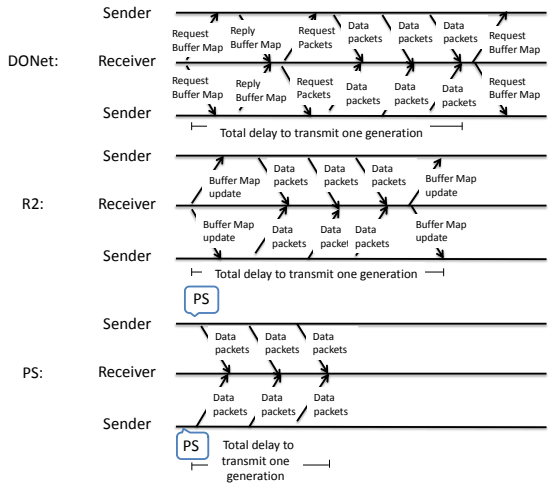

Figure 1. Comparison of the transmission delay among DoNet, R2 and the proposed PS algorithm

streams. Thomos et al. proposed a prioritized distributed video delivery with randomized network coding over a swarm-pull P2P network.[10] In their scheme, they proposed a optimized rate allocation method to delivery the live scalable video data over the Internet. The limitation of their approach is that their scheduling is based on a swarm-pull P2P network, which suffers from the problem of delay as other swarm-pull P2P networks. Sanna and Izquierdo[9] proposed a bandwidth estimation and a proactive rate selection algorithm based on the push-swarm network for scalable video streaming, but their work also needs frequently buffer-map exchange and therefore waste bandwidth. Unlike these packet scheduling algorithms, our work focuses on simplifying previous communication mechanism by estimating the number of scheduling packets in advance. The simplified transmission mechanism can work with above scheduling algorithms together to delivery scalable video over the push-swarm P2P network.

3 System Description

Building a swarm-push P2P streaming network can be mainly divided into two parts: building an overlay network, and scheduling the packet transmission. We define three types of peers in the system: *streaming source*, *tracker server*, and *client peer*. To build an overlay network, each peer contacts the tracker server to join the P2P network and form neighbour relationships. Some control information (e.g, information related to the upload bandwidth of neighbour nodes, video packets) is exchanged during the handshake procedure. To schedule the packet transmission, a push scheduling algorithm with a selection mechanism forecasts and schedules network-encoded packet transmission at each peer and a adaptive push policy works at the streaming source to improve the continuity of video playback.

Video packets at the streaming source are divided into several *generations* and each *generation* is further subdi-

vided into several blocks of 1250 Bytes. Instead of transmitting raw video packets, our P2P network transmits network-coded packets. A new network-coded packet is generated using the following equation:

$$y = \sum_{i=1}^{m} b_i c_i \tag{1}$$

where y is a coded packet. $[b_1, b_2, \cdots b_m]$ are received packets in this generation and $[c_1, c_2, \cdots c_m]$ are random coding coefficients in the Galois field of size $q = 2^8$. When a receiver receives y, it checks if the packet is innovative. If the packet is non-innovative, it is discarded. Once a client node has collected enough linear independent packets in this generation, it recovers all generations using the progressive Gauss-Jordan elimination.[9] A *streaming source PS* and a *client node PS* algorithm is performed at the source and client respectively for forecasting and scheduling. The push scheduling algorithm focuses on two aspects: 1) Find peers with uncompleted generations. 2) Decide to send the combinations of which generation.

3.1 Streaming Source Push Scheduling

The streaming source node performs the source push scheduling algorithm, which returns the scheduling results of each client node and target generation. The push scheduling policy of each generation is independent, so we focus on a generation firstly. We define the number of scheduling packets from source node to client node i in this generation is P_i. n is the number of node in the whole network. $n-2$ means $n-2$ client nodes in the network. The upload bandwidth of node i in this generation is U_i. S is the streaming rate and α is the current push factor. Then the number of scheduling packets from the source node to each peer node i is:

$$P_i = \lceil \alpha \frac{U_i}{\sum_{j=0}^{j=n-2} U_j} S \rceil \tag{2}$$

At every transmission opportunity, a selection algorithm compares the number of sent packets and the number of scheduling packets of each generation for each node. If the number of sent packets is smaller than the number of scheduling packets for a node, an encoded video packet of this generation is sent to the target client peer and the record of the number of sent packets increases. To keep the innovation over the Internet, at least one innovative packet in each generation is sent from streaming source to every client node.

3.2 Client Node Push Scheduling

We formulate the client scheduling problem as a redundancy optimization function. The main goal of our P2P push scheduling policy is to improve the bandwidth efficiency over the Internet without losing a high data delivery ratio. Then we solve the optimization problem as an integer linear programming problem using the simplex method.

The transmission of network-encoded packets at each client is periodical and is up to the Client push scheduling(PS) algorithm to decide which client node to address and for which generation to transmit. When the client node has such an opportunity, a client PS algorithm is performed. The PS algorithm of every generation is independent. The algorithm forecasts how many packets should be pushed to each receiver in each generation. We define that N_p is the number of required video packets in each generation. The number of scheduling packets from client node i to client node j is a non-negative integer H_{ij}. All H_{ij}s form matrix $\boldsymbol{H}$. N_p is the number of required packets in this generation. β is the aggressive factor to counteract the effects of the non-innovative transmission and the loss rate. R_{ij} is the number of received packets of node i from node j. We formulate it as the optimization scheduling problem with some given restrictions in the following equations:

$$\begin{aligned}
\boldsymbol{H} = \arg\min_{H_{ij}} & \sum_{i=0}^{N-2}\sum_{j=0}^{N-2} H_{ij} \text{ for } \forall i \\
\text{subject to } & \sum_{j=0}^{N-2} H_{ij} \le U_i \text{ (a)} \\
& \sum_{i=0}^{N-2} H_{ij} \ge \lceil \beta N_p \rceil \text{ for } \forall j \text{ (b)} \\
& H_{ij} \le R_{ij} \text{ for } \forall i \text{ and } \forall j \text{ (c)} \\
& H_{ii} = 0 \text{ for } \forall i \text{ (d)}
\end{aligned} \tag{3}$$

The constraint(a) means that the sum of the scheduling packets from each sender node i to other nodes should be no more than the its maximum bandwidth in this generation. The constraint(b) means that all receivers need at least $\lceil \beta N_p \rceil$ encoded packets to decode original packets. The constraint(c) means that the number of scheduling packets from i to j is smaller than the number of already received packets from j to i to guarantee an innovative transmission. The constraint(d) guarantees that a peer only send packets to other peers.

For each generation, the sender peer performs the above client PS algorithm to forecast the number of scheduling packets for each receiver. Then at every transmission opportunity of sender node, a selection algorithm compares the number of sent packets to each receiver and the number of scheduling packets calculated from the client PS algorithm. If more packets are required to sent for a generation and a receiver, the node information and the generation information are stored into a vector. For all candidates in the vector, a random one are selected and all received packets from this generation at the sender node are recombined and forwarded to the target receiver. When the window of *priority period* moves, the PS algorithm is called again for every new generation in the window. The *priority period* is a time period close to playback time and used to ensure a prioritized transmis-

sion.

3.3 Adaptive Push Policy

If a packet in the *urgent period* of a receiver node is not received, a request message is sent to the streaming source to pull this specific packet so that the video continuity of video playback is guaranteed, when some video packets are non-innovative or lost. The *urgent period* is a time period closer to playback time and is used to keep a smooth playback. The streaming source will reply this request firstly. If the source outgoing traffic rate exceeds twice the pushed packet bit rate, we double the push factor α; while if the source outgoing traffic rate goes down to 1.5 times the pushed packet bit rate, we reduce the server push factor α to half. The client PS algorithm is performed for all nodes accordingly again.

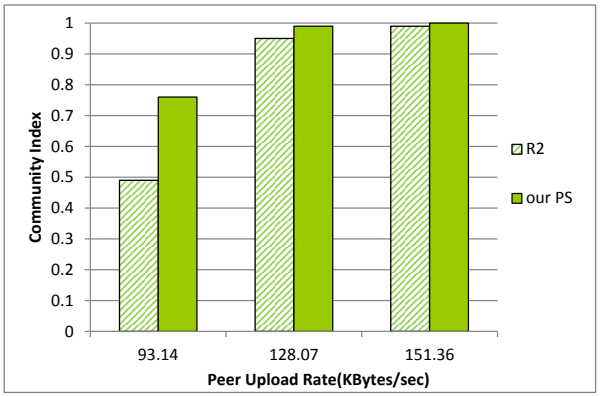

Figure 2. Performance comparison of Continuity Index between the R2 algorithm and the proposed PS algorithm

4 PERFORMANCE EVALUATION

4.1 Experiment Settings

In this subsection, the experiment settings are presented. The proposed live streaming system with push scheduling algorithm and network coding is implemented on the network simulator NS2. As commonly assumed in other P2P system studies, we simulate a P2P network where only the peer upload bandwidth is the bottleneck. In the experiment, the test network is set to be a randomly connected network. The testing video streaming is the Paris sequence encoded with H.264/AVC. The bit rate of stream S is 116 KB per second. α is setting to be 3. A generation size c is 8 frames, which corresponds to a group of pictures covering 0.26 seconds of video. The size of priority region is $8c$. We define the *playback deadline* at each peer as two seconds. The size of video blocks is 1250 Bytes. β is set to be 1.

The performances of the proposed PS algorithm are evaluated by a comparison to the other approach. The methods to be compared are random push with random network coding(R2) [13] and the push scheduling (PS) algorithm proposed in this paper. In the following experiments, only a static network is considered. The network size is setting to 20 nodes, in which the average end-to-end latency is 79ms. The urgent period is 0.5 second prio the playback time. Our evaluations focus on the following metrics: (1) Continuity Index(CI): The fraction of generation that could be decoded before the playback point. (2) The redundancy ratio: The ratio of the number of redundant video packets to the number of total received video packets. (3) Control Overhead: The number of control packets over the Internet.

4.2 Simulation Results

To evaluate the performance of the proposed P2P live streaming system, first, we compare the Continuity Index for different upload capability. As depicted in Figure 2, we compare the CI between these two methods when the peers upload rate ranging from 0.8 to 1.5 times the full rate video. The results in Fig. 2 show that the PS algorithm achieves higher or similar delivery ratio when the upload bandwidth of senders ranging from 93.14 KBytes/sec to 151.36 KBytes/sec. Compared to the R2 algorithm, the proposed PS algorithm can achieve a higher delivery ratio when bandwidth is limited.

The result in Fig. 3 shows that the proposed algorithm achieves a lower ratio of video packet redundancy when the upload bandwidth of senders ranging from 93.14KBytes/sec to 151.36KBytes/sec. The reason is that the pushing mechanism of R2 is based on the buffer-map updating. When many senders simultaneously push encoded packets to receivers, the receiver will receive many non-innovative video blocks belonging to the same generation. Comparing to the random push mechanism, the proposed PS algorithm generates less redundant video packets. With the lower redundant rate, more useful packets are pushed and the network bandwidth are fully utilized, which leads to a better CI result.

Table 1 compares the number of control packets exchanged by the peers when the upload bandwidth is 128.07 KBytes/sec and the number of peers is 20, 50 and 100 respectively, which shows a big improvement over the R2 algorithm. In our experiment, the PS algorithm generates overall fewer overheads when the upload rate and the buffer update period are the same. In all, our algorithm demonstrates a better delivery ratio and generates less redundant packets when the network is stable. The limitation of the proposed algorithm is the re-scheduling problem when the network has serious bandwidth fluctuations, which needs to be improved in the later work.

5 Conclusion

This paper proposes a novel push-based live peer-to-peer streaming system. This technique combines network coding, optimized push scheduling, a selection mechanism and

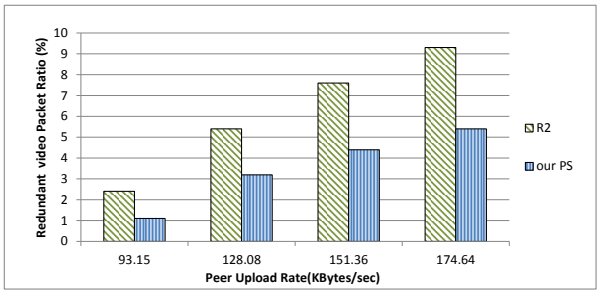

Figure 3. Performance comparison of the video packets redundancy ratio between the R2 algorithm and the proposed PS algorithm

Table 1. Comparison of the number of control packets over the Internet versus peers number

Peers Number	R2(packets/s)	PS(packets/s)
20	1538	77
50	9165	192
100	38460	397

an adaptive push policy together with the peer-to-peer communication transmission. The new approach reduces the typical source of redundant video packets and control overhead to minimum by simplifying the buffer-map updating process. The required number of packets is calculated at the sender node in advance to reach a better delivery ratio and a more efficient network bandwidth utilization. We form the push scheduling policy as an optimization problem and solve it using the linear programming algorithm. Our scheme achieves a better video delivery ratio and less volume of useless packets over the push-based P2P network, outperforming previous approaches by involving the optimized packet scheduling algorithm.

Acknowledgements

The research presented in this paper was made possible thanks to the support of the European Commission under contract FP7-ICT 611517 CONECTA 2020.

References

[1] Pplive. [Online] http://www.pptv.com/.

[2] R. Ahlswede, N. Cai, S. Y. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[3] K-HK Chan, S-HG Chan, and A. C Begen. S-panc: Optimizing scheduling delay for peer-to-peer live streaming. *Multimedia, IEEE Transactions on*, 12(7):743–753, 2010.

[4] C. Feng and B. Li. On large-scale peer-to-peer streaming systems with network coding. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 269–278. ACM, 2008.

[5] B. Li, Z. Wang, J. Liu, and W. Zhu. Two decades of internet video streaming: A retrospective view. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1s):33, 2013.

[6] Z. Liu, Y. Shen, K. W Ross, S. S Panwar, and Y. Wang. Layerp2p: using layered video chunks in p2p live streaming. *Multimedia, IEEE Transactions on*, 11(7):1340–1352, 2009.

[7] N. Magharei and R. Rejaie. Prime: Peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM Transactions on Networking (TON)*, 17(4):1052–1065, 2009.

[8] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Peer-to-peer systems IV*, pages 127–140. Springer, 2005.

[9] M. Sanna and E. Izquierdo. Proactive prioritized mixing of scalable video packets in push-based network coding overlays. *20th International Packet Video Workshop (PV)*, pages 1–7, 2013.

[10] N. Thomos, J. Chakareski, and P. Frossard. Prioritized distributed video delivery with randomized network coding. *Multimedia, IEEE Transactions on*, 13(4):776–787, 2011.

[11] D. A Tran, K. A Hua, and T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 2:1283–1292, 2003.

[12] M. Wang and B. Li. Lava: A reality check of network coding in peer-to-peer live streaming. *26th IEEE International Conference on Computer Communications(INFOCOM)*, pages 1082–1090, 2007.

[13] M. Wang and B. Li. R2: Random push with random network coding in live peer-to-peer streaming. *IEEE Journal on Selected Areas in Communications*, 25(9):1655–1666, 2007.

[14] M. Zhang, L. Sun, X. Xi, and S. Yang. igridmedia: providing delay-guaranteed peer-to-peer live streaming service on internet. *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, 2008.

[15] X. Zhang and H. Hassanein. A survey of peer-to-peer live video streaming schemes–an algorithmic perspective. *Computer Networks*, 56(15):3548–3579, 2012.

[16] X. Zhang, J. Liu, B. Li, and 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.*, 3:2102–2111, 2005.