[Proceeding] Distributed media-aware scheduling for P2P streaming with network coding

DISTRIBUTED MEDIA-AWARE SCHEDULING FOR P2P STREAMING WITH NETWORK CODING

Anooq Muzaffar Sheikh, Attilio Fiandrotti, Enrico Magli

Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Italy
anooq.sheikh@polito.it, attilio.fiandrotti@polito.it, enrico.magli@polito.it

ABSTRACT

We present a distributed packet scheduling scheme for push-based Peer-to-Peer (P2P) video streaming with Network Coding (NC) over unstructured random overlays. While previous research has shown the potentials of random-push NC for P2P, little attention has been given to the problem of scheduling the packet transmissions at the network nodes. The proposed scheduling scheme exploits the knowledge of the status of the network links and nodes to maximize the number of nodes that are able to recover the media content prior to its playout deadline. Our experiments show a large performance gain with respect to random-push scheduler in terms of better media quality.

Index Terms— Distributed Scheduling, P2P, Video Streaming, Network Coding

1. INTRODUCTION

Peer-to-peer (P2P) has emerged as an effective solution to distribute bandwidth-demanding video contents to large populations of users [1] [2]. Random Network Coding (NC) [3] can improve the performance of multicast communications as P2P video streaming, by maximizing the network throughput [4]. In NC, each network node transmits random linear combinations of the received packets to the other nodes instead of simply forwarding the received packets. Once a node has collected enough linearly independent packets, it solves a system of linear equations and recovers the message. As *any* packet collected by a node is helpful to recover the message, NC avoids the coupon collection problem typical of P2P architectures altogether. Moreover, NC enables to organize the nodes in unstructured random overlays, which require no central management and are more resilient to peer churning than the tree-based counterpart. In particular, Wang and Li [5] described a P2P streaming architecture where the peers are organized in a random unstructured overlay and operate according to a *random-push* scheduling scheme. Such scheme showed to be effective in minimizing the initial buffering times and providing almost seamless video playback with respect to network and user dynamics.

While previous research has demonstrated the advantages of random NC for P2P video streaming, comparatively little attention has been given to the problem of optimizing the packet scheduling process at the peer nodes. In [6] it has been shown that a short, non-random, pull stage after the push one can greatly help to recover from packet losses. In [7], it was shown that an appropriate packet scheduling at the network nodes can reduce the time required to recover the message, albeit this scheme considers a tree-based overlay only and NC is used for erasure-correction purposes.

In this paper, we present a distributed scheduling scheme for push-based P2P video streaming over unstructured random overlays using NC. The proposed scheduling scheme utilizes feedback information from the nodes related to packet losses and their decoding state for optimal packet transmission. In section 4, we further discuss the amount of feedback information exchanged between the nodes for the proposed scheduling scheme. The selection of the optimal scheduling policy at each transmission opportunity is formalized as an optimization problem that is independently solved at each network node. The selection of the optimal policy aims at maximizing the number of nodes that recover the message prior to a deadline keeping into account the status of the network links and nodes. Moreover, we introduce a low-complexity heuristic that further improves the performance of our scheme by improving the bandwidth allocation efficiency. Our experiments show that our scheme achieves large gains in terms of video quality over a random-push reference even when it is applied to only a fraction of the transmissions.

2. BASIC SCHEDULING IN RANDOM-PUSH P2P

In this section we overview a typical random-push P2P video streaming protocol such as [5]. The network is composed of a source node that distributes a video content to multiple cooperating nodes, organized into an unstructured random overlay. The source node holds the original video content which is divided into independently decodable chunks of data called *generations* and each generation has an associated playback deadline. Each generation is further subdivided into k symbols and encoded packets are formed from symbols belonging to the same generation. All network nodes follow a random-push packet scheduling scheme that operates as follows. At every transmission opportunity, the source node randomly en-

codes the input symbols for a given generation and produces an encoded packet that is transmitted to a node drawn randomly among those in the network. The network nodes store the received packets and the number of linearly independent packets collected by a node is called *rank* of the node. When a transmission opportunity arises for a network node, the node randomly recombines the received packets and produces a new encoded packet that is transmitted to a randomly drawn node in the network. Once a node has collected k linearly independent packets, it solves the related system of linear equations, recovers the generation and notifies the other network nodes. In general, the number k' of packets required to decode a generation is greater than k, i.e. $k' > k$ (albeit $k' \simeq k$ for a sufficiently large k), so in the following we assume that a node recovers the message after receiving k' packets.

3. PROPOSED SCHEDULING MODEL

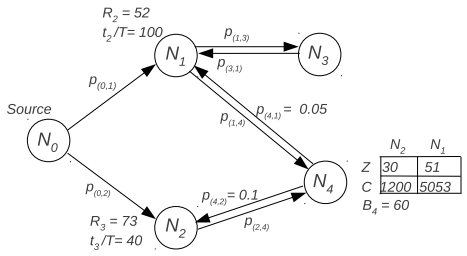

Fig. 1. Example of a network with $|V| = 5$ nodes and the parameters used in the model.

We model the network as a graph $G(V, E)$ where the vertices $V = \{\mathcal{N}_0, ..., \mathcal{N}_{|V-1|}\}$ are the nodes of the network and the arc $(i, j) \in E$ is the link that connects $\mathcal{N}_i$ to $\mathcal{N}_j$ and has associated the packet loss probability $p_{i,j}$. For any $\mathcal{N}_i$, $i \in [1, |V-1|]$, we define $A_i \subset V$ as the *neighborhood* of $\mathcal{N}_i$, that is the set of nodes that exchange packets with $\mathcal{N}_i$, where $|A_i|$ is the size of the neighborhood of $\mathcal{N}_i$. Each node $\mathcal{N}_i$ has an associated remaining budget B_i, which indicates the maximum amount of packets the node can transmit for a single generation. Every T seconds, $\mathcal{N}_i$ has the opportunity to transmits one packet and the remaining budget B_i is decremented accordingly. Each node $\mathcal{N}_j$ has an associated rank R_j, i.e. the number of linearly independent packets received so far, and a playback deadline t_j of the current generation, representing that the nodes may have a misaligned playback time. A node that achieves $R_j = k$ before the decoding deadline of the generation successfully decodes the generation and broadcasts a message to its neighbors. Each packet exchanged by the nodes contains the node state, i.e. the rank R_i of the transmitter, and the deadline t_i for the generation being decoded in addition to the payload. Figure 1 represents a sample network with the parameters associated with the network nodes and links.

Before formulating the optimization problem, we define a cost function with the goal to maximize the total number of nodes that decode the generation before the playback deadline. Let us assume that node $\mathcal{N}_i$ is given a transmission opportunity. For any $\mathcal{N}_j \in A_i$, we define $Z_{i,j}$ the expected number of packets that need to be transmitted from $\mathcal{N}_i$ to $\mathcal{N}_j$ to decode the generation, accounting for the loss probability $p_{i,j}$ as

$$Z_{i,j} = \frac{k - R_j}{1 - p_{i,j}}, \ \forall (i,j), \ i \neq j, \tag{1}$$

where $Z_{i,j} = 0$ if $i = j$. In order to account also for the decoding deadline of the generation in terms of transmission opportunities $\frac{t_j}{T}$ remaining for $\mathcal{N}_j$ to achieve full rank, we define our cost function to be minimized as

$$C_{i,j} = \frac{t_j}{T} Z_{i,j} = \frac{t_j}{T} \frac{k - R_j}{1 - p_{i,j}}, \ \forall (i,j), \ i \neq j \tag{2}$$

Finally, the optimal scheduling policy to maximize the number of nodes that recover the generation before the deadline is found solving problem in (3). The occurrence of transmission of a packet from $\mathcal{N}_i$ to $\mathcal{N}_j$ is indicated by a binary variable $x_{i,j}$, which has value 1 if the transmission does take place, and 0 otherwise. This is done by selecting $\mathcal{N}_j$ with minimum cost $C_{i,j}$, where the first summation on i is for the transmitter nodes $\mathcal{N}_i$ and second summation on j is for the recipient nodes $\mathcal{N}_j$. In particular, recipient nodes are selected among those who have not completed the generation but would still be able to, if they were served sufficient packets before their playback deadline. The dominating factor in $C_{i,j}$ is $Z_{i,j}$, which is based on the rank of the nodes. By selecting the recipient $\mathcal{N}_j$ with minimum cost $C_{i,j}$, we prioritize the node that requires less packets to decode a generation and has an earlier deadline.

$$\text{minimize} \quad \sum_{i=1}^{|V-1|} \sum_{\mathcal{N}_j \in A_i} x_{i,j} C_{i,j} \tag{3}$$

subject to

$$\sum_{\mathcal{N}_j \in A_i} x_{i,j} = 1 \ \forall i, i \neq j \tag{4}$$

$$\begin{aligned} x_{i,j} &\leq B_i \ \forall i, j \\ x_{i,j} &\leq R_i \ \forall i, j \end{aligned} \tag{5}$$

The constraint (4) means that at every transmission opportunity $\mathcal{N}_i$ can transmit one packet to one node $\mathcal{N}_j$. Moreover, the transmission occurs only if $\mathcal{N}_i$ has something to transmit ($R_i > 0$) and has some budget left ($B_i > 0$), which is represented in (5).

The problem (3) can be recast as a set of individual problems that each node solves independently as

$$\begin{aligned} &\text{minimize} \quad \sum_{\mathcal{N}_j \in A_i} x_{i,j} C_{i,j}, \ \forall i \\ &\text{subject to} \qquad (4) \text{ and } (5) \end{aligned} \tag{6}$$

since at every transmission opportunity a node transmits only one packet (4),

Finally, the problem (6) is solved at each node with the algorithm described in the following that we call DS in the rest of this work. At each transmission opportunity and for each $\mathcal{N}_j \in A_i$, $\mathcal{N}_i$ calculates the corresponding cost function $C_{i,j}$ and the $\mathcal{N}_j$ with lowest $C_{i,j}$ is selected for transmission. That is, $\mathcal{N}_i$ solves the optimization problem with a number of operations that grows linearly with $|A_i|$.

3.1. Improved Heuristic Distributed Scheduler

Since each node solves (6) independently, the same transmission policy may be selected by multiple nodes at the same time, i.e. one node may receive surplus packets than those missing to achieve full rank, resulting in a suboptimal bandwidth allocation. To address this problem, we improve the DS scheme with a heuristic which we call HDS and is described in Algorithm 1. This algorithm exploits the unique identifier of the node in the network, where the source is always $\mathcal{N}_0$ and the remaining nodes $\mathcal{N}_i$ are assigned unique i's in increasing order. First, the whole list of cost functions $C_{i,j}$ is computed and sorted in increasing order and the node at the top of the list represents the best recipient. Initially the variable *offset=0*, if $i \leq Z_{i,j}$ (line 4), then $\mathcal{N}_i$ transmits the packet to $\mathcal{N}_j$ and the algorithm ends (line 5). Otherwise $\mathcal{N}_i$ will set *offset*=$Z_{i,j}$, select the next element in $C_{i,j}$ and check the condition $i-$*offset* $\leq Z_{i,j}$ and continue the process for all A_i until it makes a transmission. The variable *offset* indicates the expected number of packets required by the previous element in $C_{i,j}$ and equally the number of transmitter nodes $\mathcal{N}_i's$ serving the node corresponding to the previous element in $C_{i,j}$. If there is no node satisfying the condition of line 4, node $\mathcal{N}_i$ randomly selects a $\mathcal{N}_j \in A_i$ that has not yet decoded the current generation, and transmits a packet to $\mathcal{N}_j$. Since $Z_{i,j}$ measures the number of packets required by $\mathcal{N}_j$ to decode the generation, the condition $i \leq Z_{i,j}$ at every transmitter node $\mathcal{N}_i$ assures that the total number of nodes that serve recipient $\mathcal{N}_j$ are not more than the packets required by recipient $\mathcal{N}_j$. In this way the number of different transmitting nodes that serve $\mathcal{N}_j$ is upper bounded by $Z_{i,j}$ and the number of surplus packets received by $\mathcal{N}_j$ is kept under control.

Algorithm 1 Heuristic Distributed Scheduler - HDS

1: Compute $C_{i,j}$ $\forall \mathcal{N}_j \in A_i$ and sort by ascending $C_{i,j}$
2: *offset*$= 0$
3: **for** each $\mathcal{N}_j$ in A_i **do**
4: **if** $(i-$ *offset* $\leq Z_{i,j})\&((Z_{i,j} > 0)\&(Z_{i,j} \leq \frac{t_j}{T}))$
5: transmit packet to $\mathcal{N}_j$ and return
6: **end if**
7: *offset*$=$ *offset* $+Z_{i,j}$
8: **end for**
9: **if** condition 4 is false $\forall \mathcal{N}_j \in A_i$
10: transmit packet to a random undecoded node $k \in A_i$
11: **end if**

4. EXPERIMENTS

In this section, we experiment with the proposed scheduling schemes using a P2P protocol similar to that we described in [8]. We stream a video sequence encoded at $C_v = 500kbit/s$ where each generation is 1 Mbit in size and is composed by $k = 100$ symbols. The problem of finding the optimal scheduling policy is independently solved for each generation. The nodes are arranged in a unstructured random overlay where the links are affected by an average packet loss rate of 10%. For the proposed DS and HDS schemes, initially, the nodes schedule the packet transmission according to a random-push scheme, during which the nodes measure the average packet loss rates $p_{i,j}$. After some initial random transmissions, the nodes switch to the optimized scheduling scheme and each node exchanges a single explicit feedback message at a random time for every 10 packets transmitted. We measure the quality of the video recovered at the nodes in terms of Continuity Index (CI), which is defined as the fraction of generations that could be decoded prior to the playback deadline.

First, we evaluate the video quality as a function of the total upload bandwidth in the network for a network composed of $|V| = 100$ nodes where the nodes are arranged in a fully connected overlay. Initially, we set the upload bandwidth of the network nodes C_n to match the video bandwidth, i.e. $C_n = C_v$, then we gradually increase C_n up to 1.5 times the video bandwidth, i.e. up to $C_n = 1.5C_v$. We experiment with the DS and the HDS schedulers presented in Section 3 plus two reference schemes. The first reference is a simple random-push scheduler as described in Section 2. The second reference is an oracle scheme, where we assume that an omniscient central coordinator that knows the state of each node optimizes the scheduling of all the network nodes.
Figure 2 shows the results of the experiments. When $C_n = C_v$, the CI achieved by DS and HDS is 0.72 and 0.90 respectively, while the CI achieved by the random-push reference is just 0.18. The random-push scheduler does not take into account the decoding status, i.e. the rank, of the nodes nor the playback deadlines. So, nodes that have already recovered a generation receive surplus packets, reducing the transmission budget available for the other nodes. Even the Oracle scheme achieves a CI below 1.0 as the total upload bandwidth in the network is lower than the minimum bandwidth required to cope with the losses on the links. When the upload bandwidth increases to 1.1 times C_v, the CI achieved by random-push, DS and HDS is about 0.50, 0.81 and 0.99 respectively. That is, the HDS scheduler achieves a CI close to 1 when the extra upload bandwidth available in the network is just enough to compensate the losses on the links. By comparison, the random-push and DS schedulers require that the overall upload bandwidth is 1.35 and 1.45 times C_v to achieve a CI

close to 1. Furthermore, the HDS and Oracle curves almost overlap, showing that the HDS scheme is effective in allocating the output bandwidth.

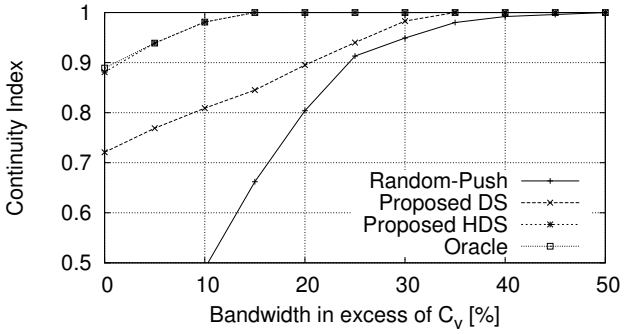

Fig. 2. Continuity Index as a function of upload bandwidth.

Second, we evaluate the video quality for networks of different size $|V|$ and for an overlay where the size of neighborhood of the nodes is $\frac{|V|}{4}$. The output bandwidth of the nodes is set to $C_n = 1.15C_v$, as the previous experiments showed that it is enough for the HDS scheme to achieve a CI close to 1. Figure 3 shows the CI obtained by the DS and HDS schemes plus the random-push reference. The CI achieved by the random-push reference never exceeds 0.6, while it slightly increases from 0.55 for $|V| = 25$ to 0.6 for $|V| = 200$. As $|V|$ increases, the probability of a node being selected for the transmission decreases, so fewer surplus packets are transmitted increasing the transmission budget for nodes that have not yet decoded the generation. The DS scheduler performs better than the random-push reference, however the CI never exceed 0.9 due to some surplus transmission that it cannot avoid. Finally, the HDS scheduler achieves a CI close to 1 by avoiding almost entirely the transmission of surplus packets.

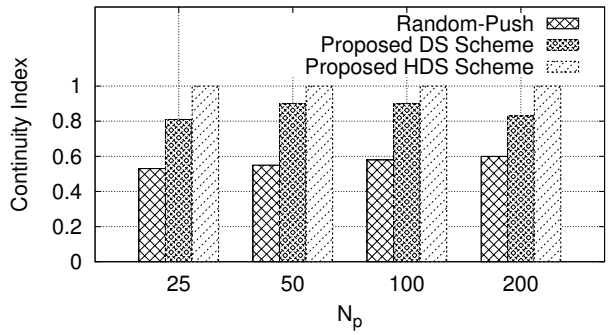

Fig. 3. Continuity Index as a function of the number of network nodes.

Third, we evaluate the video quality as a function of the number of initial random-push transmissions and the size of the neighborhood for the HDS scheme. We are interested in investigating how many random-push transmissions are allowed and how small the neighborhood of a node can be before the video quality starts to degrade. From a computational complexity perspective, random-push transmissions are more desirable because they reduce the number of times Algorithm 1 is executed by the nodes. Similarly, a small neighborhood is desirable because it reduces the number of the cost functions $C_{i,j}$ that are computed at each execution of Algorithm 1 plus the number of explicit feedback messages broadcasted by the nodes. Figure 4 shows the results of the experiments for a network composed of $|V| = 200$ nodes. For a fully connected mesh overlay ($A_i = |V|$), the CI is close to 1 if the number of random transmissions does not exceed 93% of the total, i.e. if at least 7% of the transmissions are optimized using the DHS scheme. However, when the neighborhood size is reduced to $\frac{|V|}{4}$, the maximum number of random transmission allowed before CI drops below 1 decreases to 85%. As the neighborhood size decreases, the nodes have in fact fewer neighbors and so the probability of transmitting surplus packets during the random-push stage increases, resulting in a less efficient bandwidth allocation.

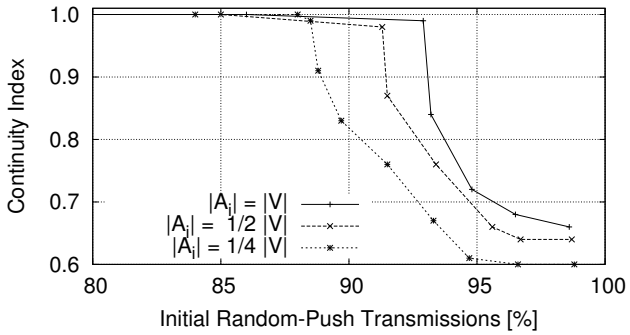

Fig. 4. Continuity Index as a function of initial random-push stage for different neighborhood sizes $|A_i|$ for the proposed HDS scheme.

5. CONCLUSION

We have presented an optimized scheduling scheme for push-based P2P streaming using NC, implemented by means of heuristic algorithms in a fully distributed way. We compared the performance of our scheduling schemes with a random-push and an omniscient oracle references measuring the quality of the video received at the nodes. Our experiments show that the proposed scheduling scheme constantly outperforms the random-push and performs close to the oracle reference thanks by efficiently allocating the output bandwidth available at the nodes. Moreover, our experiments suggests that less than 10% of the transmissions need to be optimized, thereby with little extra computational complexity with respect to the random-push reference.

6. ACKNOWLEDGMENTS

Part of this work was supported by PRIN ARACNE, a research project funded by the Italian Ministry of Education and Research.

7. REFERENCES

[1] "Sopcast, http://www.sopcast.com," .

[2] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y.-S.P. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, March 2005, vol. 3, pp. 2102 – 2111.

[3] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: an instant primer," *Computer Communication Review*, vol. 36, no. 1, pp. 63, 2006.

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

[5] Mea Wang and Baochun Li, "R2: Random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1655 –1666, december 2007.

[6] M. Toldo and E. Magli, "A resilient and low-delay p2p streaming system based on network coding with random multicast trees," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2010, pp. 400–405.

[7] K.-H.K. Chan, S.-H.G. Chan, and A.C. Begen, "Spanc: Optimizing scheduling delay for peer-to-peer live streaming," *Multimedia, IEEE Transactions on*, vol. 12, no. 7, pp. 743 –753, nov. 2010.

[8] A. Fiandrotti, A. M. Sheikh, and E. Magli, "Towards a p2p videoconferencing system based on low-delay network coding," in *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012, pp. 1529 –1533.