Live Peer-to-Peer Streaming with Scalable Video Coding and Networking Coding

Shabnam Mirshokraie
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

ABSTRACT

We present the design of a peer-to-peer (P2P) live streaming system that uses scalable video coding as well as network coding. The proposed design enables flexible customization of video streams to support heterogeneous receivers, highly utilizes upload bandwidth of peers, and quickly adapts to network and peer dynamics. Our design is simple and modular. Therefore, other P2P streaming systems could also benefit from various components of our design to improve their performance. We conduct an extensive quantitative analysis to demonstrate the expected performance gain from the proposed design. Our analysis uses actual scalable video traces and realistic P2P streaming environments with high churn rates, heterogeneous peers, and flash crowd scenarios. Our results show that the proposed system can achieve: (i) significant improvement in the visual quality perceived by peers (several dBs are observed), (ii) smoother and more sustained streaming rates, (iii) higher streaming capacity by serving more requests from peers, and (iv) more robustness against high churn rates and flash crowd arrivals of peers. This paper shows that the integration of network coding and scalable video coding in P2P live streaming systems yields better performance than current systems that use single-layer streams and proposed systems that use either network coding alone or scalable video coding alone.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Design

Keywords

Peer-to-peer streaming, scalable video coding, network coding, live streaming

MMSys'10, February 22–23, 2010, Phoenix, Arizona, USA.

1. INTRODUCTION

Peer-to-peer (P2P) live video streaming systems have seen wide deployment around the globe. P2P streaming systems such as CoolStreaming [41], PPLive [29], UUSee [34], SopCast [32], and TVAnts [33] attract numerous users every day. As more users get used to viewing multimedia content online, they will demand higher and better video quality than available on many of the current P2P streaming systems. As an indication of this demand and the response from industry, Huang et al. [14] show that the average bit rate of videos offered by the MSN Video Services has increased by more than 50% over a nine-month period, and it is the likely that the bit rate will continue to increase in the future. Providing high-quality streaming over P2P systems, however, faces multiple challenges, including: (i) limited upload capacity of peers, (ii) high heterogeneity of receivers in terms of download bandwidth, screen resolutions, and CPU capacity, and (iii) high churn rate as the peer population is constantly changing. Addressing these challenges requires not only increasing the capacity of peers and deploying additional seeding servers to make up the shortage in resources, but also employing novel methods for encoding and distributing multimedia content and developing algorithms and protocols to optimally utilize the available resources.

In this paper, we propose a new design for P2P live streaming systems that, we believe, will significantly improve their performance. The new design strives to address many of the challenges impeding current systems by efficiently utilizing peers' resources, easily customizing multimedia content to support receivers with diverse resources and requirements, and quickly adapting to network and peer dynamics. Our design is simple and practical; we actually have implemented it. Our design employs scalable video coding to support heterogeneous receivers as well as networking coding to maximize the streaming throughput and handle network dynamics. Although scalable video coding and network coding have been *individually* proposed for various systems in the literature, e.g., [10, 11, 13, 18, 24, 28], their *integrated* use in P2P live streaming systems has not been fully explored in the literature, to the best of our knowledge. The integration of these two technologies provides many performance benefits beyond those offered by each of them individually, as will be shown in this paper.

We present the design of a P2P streaming system that employs both scalable video coding and network coding. Our design is modular and can be used as an improvement plug-in in other P2P streaming systems. That is, we focus on the new components needed to handle multimedia content

compressed in scalable manner and encoded using network coding. Thus, our work and software components are readily useful for other systems. In addition, we *quantitatively* show the expected performance gain from the proposed design using actual scalable video traces in realistic P2P streaming environments with high churn rates, heterogeneous peers, and flash crowd scenarios. In particular, our results show that the proposed system can achieve (i) significant improvement in the visual quality perceived by peers (several dBs are observed), (ii) smoother and more sustained streaming rates (up to 100% increase in the average streaming rate is obtained), (iii) higher streaming capacity by serving more requests from peers, and (iv) more robustness against high churn rates and flash crowd arrivals of peers.

The rest of this paper is organized as follows. In Section 2, we provide a brief background on network coding and scalable video coding, and we summarize the previous work in the literature. In Section 3, we describe the proposed system. In Section 4, we evaluate the proposed system using actual video traces, and we conclude the paper in Section 5.

2. BACKGROUND AND RELATED WORK

2.1 Brief Background

We present a brief overview of network coding and scalable video coding, which are employed in the proposed P2P streaming systems. More details about network coding can be found in [1, 5, 8] and the references therein, and more information about scalable video coding can be found in [31] which describes the recent H.264/SVC standard.

Network Coding. In traditional packet forwarding methods, each node simply repeats data packets destined to other nodes in the network. In contrast, the network coding concept enables source and intermediate nodes to perform simple operations on packets before forwarding them. These operations allow nodes to send partial information to the destination. After receiving all the necessary partial information, the receiver will be able to recover the original packet. Packing information at intermediate or source nodes is called encoding and extracting real data from encoded ones is referred to as decoding. Encoding and decoding are linear operations over a Galois Field of size 2^l, which is denoted by GF(2^l). A GF(2^l) is a finite field in which operations are done on l bits of data.

Encoding is a linear combination of blocks, which is formulated as: $x = \sum_{i=1}^{n} c_i \cdot b_i$, where n is the total number of blocks, c_is are coefficients of size l taken from GF (2^l) and b_is are data blocks of size k bytes. The symbol x represents one encoded block of size k. Each encoded block is a linear combination of the original blocks. Thus, it is uniquely identified by the set of coefficients included in the linear combination. Multiplications and additions are done over GF (2^l).

Assuming a node receives set of (C^1, x^1), (C^2, x^2), ..., (C^m, x^m), where C^i is the vector of coefficients $(c_1^i, c_2^i, \ldots, c_m^i)$ for block x^i. The decoding process is performed by solving: $x^j = \sum_{i=1}^{n} c_i^j \cdot M^i$, where M^is are unknowns. This is a system of linear equations with n unknowns and m equations, which can be solved by the Gaussian elimination method if $m \geq n$. However, it is not necessary to receive all equations in order to decode all blocks. Usually some blocks can be recovered before receiving all encoded blocks.

Scalable Video Coding. The recently H.264/SVC video coding standard [31] adds scalability to the widely used H.264/AVC video coding technique [38]. The H.264/SVC standard supports temporal, spatial, and quality scalability at the same time. Temporal scalability is achieved by employing a hierarchical prediction structure among video frames belonging to the same Group-of-Pictures (GoP), in which frames of higher temporal layers can only be predicted from lower temporal layers. In the spatial scalability, a spatial layer s of a frame can be predicted from the s-th spatial layer of some other frames (in lower temporal layers), as well as lower spatial layers in its own frame. For providing quality scalability, there are two possibilities. The first one follows the spatial scalability structure, but assigns the same resolution and different quantization parameters to layers. This produces a Coarse-Grained Scalable (CGS) video with limited number of quality layers. A finer granularity can be provided by the second possibility, which uses Medium-Grained Scalability (MGS) coding to divide a single CGS quality layer into multiple sub-layers, which are referred to as MGS layers. This is done by partitioning the residual DCT coefficients of a CGS layer into multiple MGS layers. A stream can be truncated at any CGS or MGS layer. In addition, some packets of an MGS layer can be discarded, while the remaining ones can still be decoded to improve quality. Packet discarding can be done in arbitrary ways, depending on the bitstream extraction process [2]. H.264/SVC allows Up to 7 temporal, 8 spatial, and 16 quality layers [31]. Using scalable video coding, users with high link capacities experience better video quality by receiving more layers, while others with lower bandwidth get quality proportional to the number of layers they can receive.

2.2 Related Work

P2P streaming systems are often divided into two major categories: tree-based and mesh-based (also known as swarm-based and data driven). In tree-based systems, peers organize themselves into one or more multicast trees and data will be pushed along the tree structure [3, 16]. Tree-based systems incur high costs for the management and maintenance the tree structure, especially with high peer churn rates. Mesh-based systems allow peers to self-organize in mesh-shaped graphs [21, 40, 41]. These systems usually yield better performance in practice as they are more robust against high-level of peer and network dynamics [22]. Our work focuses on mesh-based P2P streaming systems.

Most of the currently deployed P2P streaming systems, e.g., [29, 32–34, 41], use nonscalable video streams. Thus, they serve a single-version of the video stream to all peers, and they have limited support for heterogeneous peers. To address these issues, number of works have proposed P2P streaming systems with scalable video streams, e.g. [6, 12, 17, 24, 30]. Cui and Nahrstedt [6] present an algorithm to decide for each peer how to request video layers from a given set of senders. They assume layers have equal bitrate and provide equal video quality. Hefeeda and Hsu [12] study this problem for Fine-Grained Scalable (FGS) videos, taking into account the rate-distortion model of the video for maximizing the perceived quality. Rejaie and Ortega [30] present a framework for layered P2P streaming, where a receiver coordinates the transmission of video packets from multiple senders using a TCP-friendly congestion control mechanism. Lan et al. [17] propose a scheduling algorithm for peers to request data from senders. The allocation of seed server re-

sources in P2P streaming systems with scalable videos has also been considered in [24]. While these works enable serving streams with different qualities to peers with diverse resources, none of them employs network coding to further enhance the utilization of peer resources.

Network coding has been shown to maximize the throughput and bring various performance benefits in different environments [1, 15, 19]. For example, in wireless networks, network coding improves the message delivery probability for ad-hoc multicast protocols [27], and overcomes broadcast storm problems [25]. Network coding has also been proposed for P2P file-sharing systems. For example, in the Avalanche system [9, 10], the authors use randomized network coding to efficiently distribute files and to decrease the download time. The authors provide a method to ensure that any piece uploaded by a peer can be useful to other peers. However, these techniques are not applicable to P2P streaming systems, which have strict timing constraints and packet sequence requirements.

Several works have proposed using network coding for P2P streaming applications, including [36,37], where the authors address practical aspects of using network coding in such systems. Feng and Li [7] develop analytical models to show the benefits of using network coding in live P2P streaming systems. All of these works confirm the viability of network coding in different applications. However, none of them has considered integrating network coding with scalable video coding to support wider ranges of clients. They basically improve the performance of single-layer P2P streaming systems.

Recently, a few works have considered both network coding and scalable video streams [4,42]. For example, Zhao et al. [42] try to provide each end user in a multicast session with the maximum number of layers through solving an optimization problem using a greedy algorithm. While in [4], Chenguang et al. formulate an integer linear programming (ILP) problem to solve the same problem. Unlike our work, the above works target tree-based P2P streaming systems, and they assume that peers know the global tree structure and this structure is fairly static. These assumptions typically do not hold in practice. In contrast, we target the highly dynamic mesh-based P2P streaming systems with no assumptions/constraints on the topology.

Finally, Nguyen et al. [26] propose hierarchical network coding (HNC) to be used with scalable video coding. HNC performs network coding across all layers of the same video stream to provide higher error protection to lower video layers. HNC is designed to reduce the impact of packet losses. However, it assumes that most users are capable of or willing to receive all video layers. For example, a limited bandwidth receiver that is interested in only 2-layer version of the stream may end up receiving data blocks from higher layers, although the data cannot be used. Thus, the bandwidth of peers can be wasted. This implies that HNC will not efficiently support heterogeneous clients. In addition, performing network coding on all layers will increase the size of the coefficient matrix needed for network coding operations. Since the time and space complexities of the encoding and decoding processes depend on the size of the coefficient matrix, HNC will impose a significant overhead on peers, which have limited-resources in the first place. Furthermore, the work in [26] did not provide a rigorous quantitative evaluation of HNC in real dynamic P2P environments, as we do.

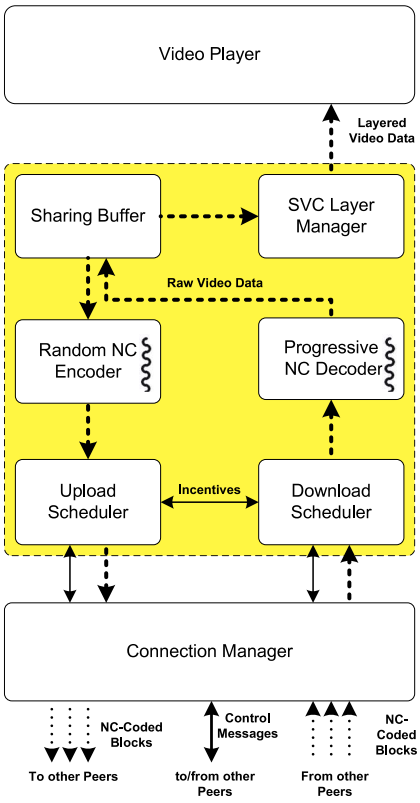


Figure 1: Peer Software Architecture. Dashed arrows denote video data, and solid arrows denote control messages.

Our proposed system performs intra-layer network coding and is fairly efficient.

3. PROPOSED P2P STREAMING SYSTEM

In this section, we describe the proposed P2P live streaming system that employs network coding and scalable video coding. We start with a high-level overview, followed by more details.

3.1 Overview

We target mesh-based P2P streaming systems which have been widely used in practice [21, 40, 41]. In our system model, there are three entities: tracker, source, and peer. The tracker matches peers who are viewing the same video stream. This matching results in multiple dynamic swarms in the system. There is at least one source node in the system to introduce the original streams to peers. The source node (sometimes called seed server) also provides additional capacity in case that peers do not have enough resources and in the beginning of the sessions where very few peers exist. Source nodes perform network coding operations on the scalable video streams in order to prepare them for distribution in the system. Peers act as receiving clients as well as share some of their upload bandwidth to serve other peers. As receivers, peers decode network-coded data received from others and process this data to create proper scalable video streams and to ensure smooth video quality. As senders, peers encode video data using network coding

with parameters based on their own upload capacity as well as the characteristics of the receiving peers.

A simplified model for the software architecture of a peer in our system is shown in Fig. 1. A similar model is used for source nodes, but with some differences as elaborated later. We do not address the design or optimization of trackers; the function of the tracker is orthogonal to the work presented in this paper. We also do not address other problems in mesh-based P2P streaming systems, including neighbor selection, gossip protocols (for exchanging data availability), incentive schemes, and overlay optimization—which all have been heavily researched in the literature. All of the above issues are abstracted in the Connection Manager component in Fig. 1, while our work is focused on the components in the shaded box in that figure. The separation and abstraction of functions enable us to support different P2P streaming systems with minimal changes in our design and code. Therefore, our work is fairly general.

3.2 Details

Peer Software Architecture and Functions. The main functions of a peer in our system are summarized in Fig. 1. We first describe the receiving part of the peer model.

At the receiver side, a peer interested in receiving a specific part of the video stream, determines and requests a proper number of encoded blocks through the Download Scheduler. The Download Scheduler computes the number of required encoded blocks based on the current available bandwidth and the number of video layers that the receiver is interested. It then assigns each of the active senders in the session blocks to send proportional to its upload bandwidth.

Immediately after receiving any encoded block through the network, the block is forwarded to the Progressive NC Decoder component. The Progressive NC Decoder rearranges the coefficients and encoded block matrices. This is done through one round of the Gauss-Jordan elimination method [37]. The Gauss-Jordan elimination method is a version of the Gaussian elimination method which inserts zeros above and below the pivot elements in the matrix as it goes from the top row to the bottom one. In other words, the Gauss-Jordan elimination method converts a matrix to its reduced row echelon form (RREF) where every leading coefficient is 1 and it is the only nonzero element in its column.

Using the Gauss-Jordan elimination method allows the process of decoding to start before receiving all encoded blocks [37]. It also enables the Progressive NC Decoder to immediately remove any dependent linear equations, as it converts all the coefficients of a received dependent linear row in the coefficient matrix into zeros. This signals the Progressive NC Decoder to eliminate this row from the coefficient matrix immediately after it is received. The Progressive NC Decoder will then investigate the coefficients matrix. If it is reduced to an identity matrix, the resulted encoded blocks are equal to the original blocks without any further decoding process. If the original data is obtained, it will passed to the SVC Layer Manager, which prepares the video data for the video player. After a block is successfully decoded, it will be stored in the Sharing Buffer for potential upload to other peers.

Next, we describe the uploading part of the peer model in Fig. 1. Network coding enables senders to provide receivers with partial information without needing a huge buffermap to keep availability of each partial data. The mechanism for producing such kind of partial data is as follow. Upon receiving a request at the sender side, random network coding is performed on the blocks of the requested layer. Random network coding is used because it provides robustness against frequent network topology changes, and it eliminates the need for having a centralized knowledge about the network topology [5]. In random network coding, the encoding process is done through randomly and uniformly selecting coefficients from the Galois Field. It has been proved that by using random network coding even with a small Galois Field size, such as 8, the probability of selecting linearly dependent combinations is negligible [39]. In order to reduce the network coding complexity, we need to reduce the number of blocks [23]. For this purpose, in our network coding scheme, we apply network coding operations on blocks of each video layer separately. Furthermore, we use a Galois Field of size 8, as larger sizes increase the complexity while improving the results only marginally.

Table 1: Characteristics of Videos used in the Experiments.

Video Trace	Sony Demo	Tokyo Olympics	NBC News
Average PSNR (dB)	47.6	42.7	35.5
Average Bit Rate (kbps)	850	500	325

Source Node Software Architecture and Functions. A source node prepares video streams before introducing them into the system. A video stream is encoded into multiple layers in a scalable manner. The video stream is divided into equal-length segments. Each segment contains a fixed number of video frames, e.g., 30 frames. Since we consider scalable streams, each video frame is composed of multiple layers. We apply network coding operations on the data contained in individual segments as follows. The video data of each layer in a segment is divided into fixed-size blocks. Then these blocks are encoded. Notice that different layers may contain different number of blocks, depending on the visual complexity of the video frame.

The encoding process is applied at the source nodes by using random network coding. On intermediate nodes, i.e., uploading peers, the blocks are re-encoded with different coefficients. In both cases, the coefficients of each block are attached to the block itself during transmission.

Overhead Analysis. There are two kinds of overhead imposed by the proposed streaming system: communication and computation. The communication overhead is due to attaching the encoding coefficients to the encoded data blocks. In practical applications, the size of the coefficients is small compared to the block size. The computation overhead is imposed by the encoding and decoding processes of the network coding scheme. These processes require quadratic number of operations in terms of the number of blocks in a segment. These operations are on finite fields and thus are performed as xor operations, which can be done efficiently by the processor. Therefore, most of the current commodity PCs can easily handle the encoding and decoding operations.

Table 2: Upload Bandwidths Distribution of peer.

Fraction of Peers (%)	10.0	14.3	8.6	12.5	2.2	1.4	6.6	28.1	16.3
Total Bandwidth (kbps)	256	320	384	448	512	640	768	1024	> 1500
Contributed Bandwidth (kbps)	150	250	300	350	400	500	600	800	1000

4. EVALUATION

4.1 Experimental Setup

We have implemented the proposed P2P streaming system in Java. Our implementation performs all functions described in Section 3 and summarized in Fig. 1. Our implementation was validated by using actual video streams.

To conduct rigorous quantitative analysis of the proposed system under wide range of working conditions, we implemented a testing application to emulate the characteristics of realistic P2P streaming systems. This testing application enables us to conduct controllable and repeatable experiments with different parameters and large number of peers. We considered deploying our system on the PlanetLab testbed and on our own local area testbed. However, these testbeds would not allow us to control important parameters such as peer upload/download bandwidth, neither would they enable us to test under high churn rates, flash crowd scenarios, and large number of *heterogeneous* peers.

The setup of our experiments is as follows. We use multiple scalable video traces obtained from the Arizona State University video library [35]. In particular, the results in this paper are based on three video streams: a demonstration from Sony, a clip from the Tokyo Olympics and a clip from NBC News. These videos are chosen because they have diverse characteristics in their quality and bit rates. This diversity is important to assess the performance of our system in real settings. Each video is encoded in 5 scalable layers and has a frame rate of 30 fps. The frame size is CIF (352x288) and each group-of pictures (GoP) has 16 frames. We use 10 minutes of each clip in our experiments. Table 1 summarizes the characteristics of these video streams.

We divided each video stream into segments, where the segment size is varied. Each layer of a segment is then encoded using network coding in a number of blocks. We use different block sizes for evaluating the performance of network coding. But in any given experiment, the block size is fixed for all layers in a segment and all segments in the video. This is done to reduce the computation complexity of the network coding process, as network coding with variable block sizes is expensive. Since the video visual content changes with time, the number of blocks in a segment varies with the size of the video frames in that segment.

We create a highly-dynamic P2P streaming system with more than 1,000 heterogeneous peers that are continually changing. The upload bandwidth values of peers are chosen according to the distribution given in Table 2. This distribution is recommended based on actual measurement studies performed in [20]. Peers in our system can randomly fail, and they join/leave the system following different probability distributions, where each probability distribution is chosen to create a specific testing scenario such as flash crowd arrivals and high peer churn rates. More details will be given in the corresponding experiments later.

We compare the proposed system (denoted by SVC+NC in the figures) against three different systems: (i) a system that uses scalable video coding only (denoted by SVC), (ii) a system that uses single layer video streams with network coding (denoted by SL+NC), and (iii) current systems that use single layer, nonscalable, streams (denoted by SL). We consider several performance metrics, including: (i) average streaming rate, (ii) average streaming quality, (iii) number of streaming requests served, and (iv) fraction of late frames. These performance metrics are computed across all peers in the system, for diverse video streams, under various network conditions, and different probabilistic distributions for peer behavior. Moreover, most of these metrics are computed on a frame by frame basis and consider each layer in every frame. Therefore, we believe that our experiments are comprehensive and the results are representative of real systems.

4.2 Results

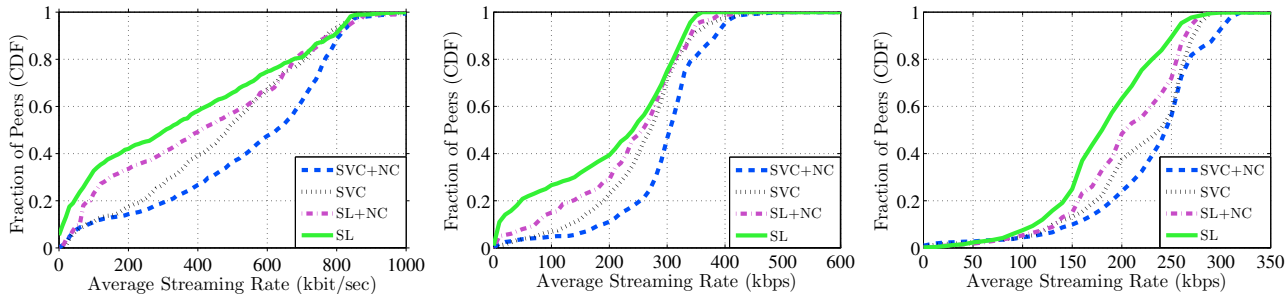
We present the results of our extensive evaluation in the following. We first present the results for the performance metrics mentioned above. Then, we analyze the impact of several system parameters on the performance and robustness of the proposed system, specially in presence of high peer churn rates, flash crowd arrivals, and when different segment and block sizes are used.

Average Streaming Rate. We measure the average streaming rate during live streaming sessions. We define the streaming rate as the total amount of received video data per second. The average streaming rate is computed across all active peers and represents a basic performance metric.

We schedule 1,000 peers to uniformly at random join the P2P streaming system during the 10-min simulation time. We also schedule a faction of the peers to fail or depart the system uniformly at random during the simulation time. On overage, 10% of the peers leave the system at random times. For each streaming session, a receiver is randomly chosen. Then, a group of five senders that already have the requested stream are randomly chosen to serve the receiver.

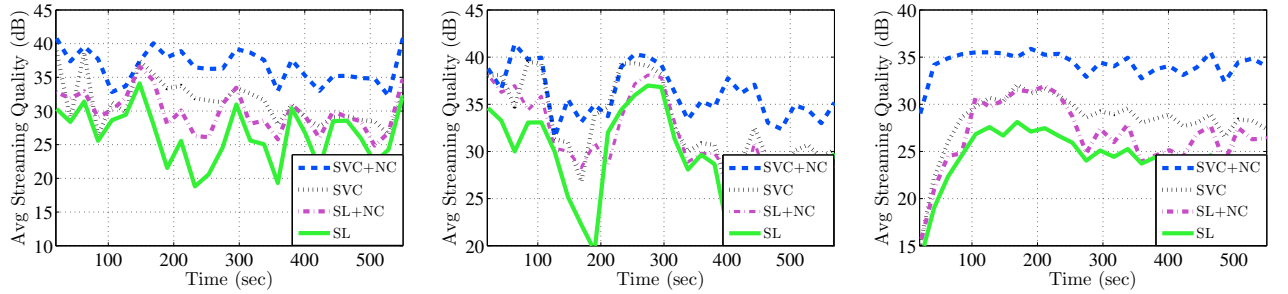
We plot the results for three different video clips in Fig. 2 as a CDF (cumulative distribution function). The figure clearly shows that the proposed SVC+NC system outperforms the other three systems by a wide margin. For example, in the Sony Demo video, less than 18% of the peers in our SVC+NC system obtain a streaming rate of 200 kbps or less, while more than 40% of the peers in the current single layer streaming systems receive at that low rate. On the other hand, almost 50% of the peers in our SVC+NC system receive a high streaming rate of at least 600 kbps, while this percentage is only about 30% in SVC and SL+NC, and 22% in SL systems.

Average Streaming Quality. Next, we consider the video quality for each active peer. Unlike the average streaming rate, which is a raw performance metric, the video quality depends on the characteristics of the video streams being served in the system and it is closer to the actual quality perceived by users. There are several methods for computing



(a) Sony Demo (b) Tokyo Olympics (c) NBC News

Figure 2: Average streaming rate achieved using different systems.



(a) Sony Demo (b) Tokyo Olympics (c) NBC News

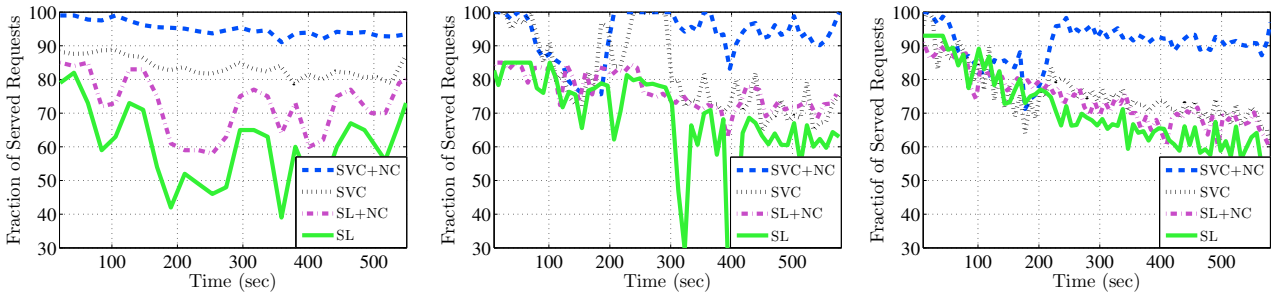
Figure 3: Average streaming quality achieved using different systems.

the video quality. We choose the Y-PSNR (peak signal to noise ratio of the intensity component of the video) as our video quality metric, because it is simpler to compute and interpret by readers. We acknowledge that the Y-PSNR may not always be the most accurate quality metric for all videos, but it is sufficient for comparative study in this paper.

We compute the quality as the Y-PSNR of the frames received on time divided by the total number of frames. Then we take the average among all peers and plot the results in Fig. 3. The figure shows that the average quality in the SVC+NC system is consistently higher than that in the SVC, SL+NC, and SL systems. For example, for the NBC News video in Fig. 3(c), the SVC+NC system yields up to 10 dB improvement in quality compared to the current single layer streaming systems. This is a substantial gain that will definitely be felt by users and will increase their satisfaction from the P2P streaming system. Fig. 3(c) also shows that the proposed SVC+NC outperforms the SVC and SL+NC systems by up to 5 dB, which is also a significant gain. The results for other videos indicate similar gains. In addition, the results in Fig. 3 indicate that the video quality achieved by the SVC+NC system is more stable and smooth over time. For example, in Fig. 3(b), there is a dramatic drop in quality for the SL system around 200 sec of the video because of the increased visual complexity of the video in this period. In contrast, the SVC+NC system did not suffer a large drop in quality.

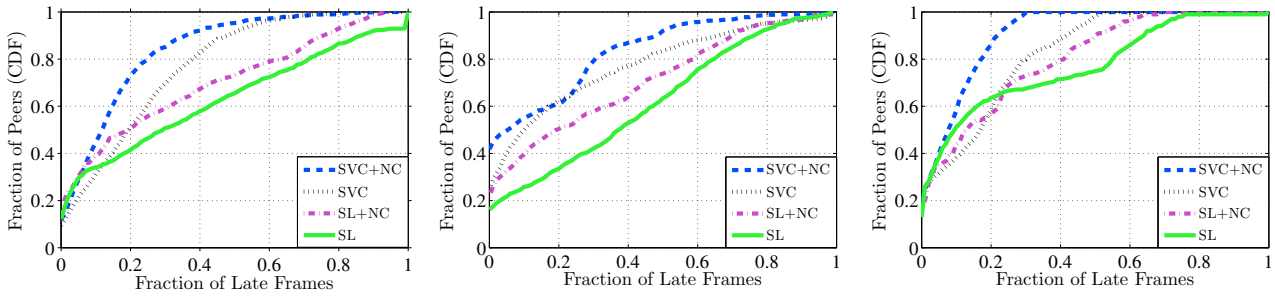
In addition to the average video quality, we measure the fluctuations in the video quality. We quantify the fluctuations by measuring the standard deviation of the observed quality with time. We then compute error bars defined by three points: average quality minus one standard deviation, average quality, and average quality plus one standard deviation for each peer. We do not plot these error bars in Fig. 3 because they clutter the figure. Our results show that the proposed system provides much smoother quality for peers than other systems. In particular, the quality fluctuations in the proposed SVC+NC system are about 100% less than the fluctuations observed in the current single layer streaming systems, and about 50% less than the fluctuations in the other SL+NC and SVC systems.

In summary, the results for the above two metrics (average streaming rate and quality) demonstrate that the proposed SVC+NC system outperforms the other systems in both raw as well as user-perceived performance dimensions. The reasons for this better performance can be summarized as follows. Single layer streaming systems are not flexible in terms of adapting the quality to the current network and peer conditions. They also do not provide optimal throughput. Therefore, they yield the worst performance. Streaming systems that use network coding with single layer videos increase the system throughput, but do not improve the flexibility of the single layer video streams. Thus, SL+NC systems improve the performance beyond what is achievable by SL systems. On the other hand, P2P streaming systems that use scalable video streams adapt well to network and peer dynamics, but they may not fully utilize peer resources. Therefore, SVC systems also improve the performance compared to SL systems. Combining scalable video streams with network coding achieves both flexibility and increased throughput. Thus, SVC+NC systems consistently provide superior performance compared to other systems.



(a) Sony Demo (b) Tokyo Olympics (c) NBC News

Figure 4: Number of served requests for different systems.



(a) Sony Demo (b) Tokyo Olympics (c) NBC News

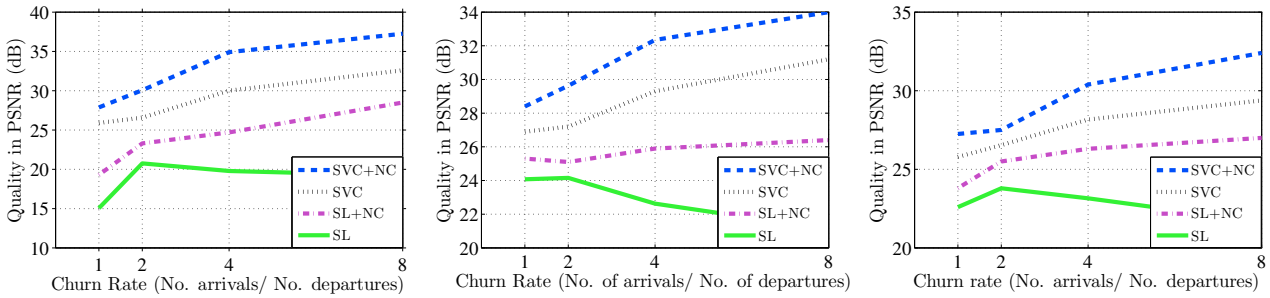
Figure 5: Fraction of late frames for different systems.

Number of Streaming Requests Served. In Fig. 4, we plot the fraction of served requests in different streaming systems. We refer to a request as served when it is responded by neighbors and received on time by the requesting peer. We obtain fraction of served requests by computing the number of served requests made by active peers divided by the total number of requests. We compute this fraction every 20 seconds. The results in Fig. 4 show that the proposed SVC+NC system serves more requests than the other systems. For example, for the NBC News video in Fig. 4(c), up to 30% more requests can be served using the SVC+NC system than the SL system. Therefore, the proposed system not only provides better video quality, but also serves more requests from peers.

Fraction of Late Frames. Next, we analyze the fraction of late frames for all considered streaming systems. The fraction of late frames is obtained by dividing the number of late frames to the total number of requested frames. When a peer first joins the network, it waits for a few segments of the video according to its initial buffering delay. The initial buffering delay helps peers to maintain synchronized playback and cooperate with each other effectively [7]. In all experiments, we let peers wait for two segments when they join the system as recommended by [7]. We plot the CDF of the late frames in Fig. 5. The figure shows that in the SVC+NC system, more frames meeting their deadlines than in the other systems. For example, in Fig. 5(a), in the single layer streaming system, about 16% of the peers received more than 80% of the frames after their deadlines. While in the proposed SVC+NC system, almost no peer observed that high fraction of late frames. As another example, Fig. 5(b) shows that the SVC+NC system achieves nearly 100% improvements over other systems in terms of the fraction of peers that observed no late frames: from about 20% of the peers in the SL, SL+NC, and SVC systems to about 40% in the SVC+NC system. Finally, Fig. 5(c) shows that there is no peer with more than 30% of late frames, while this fraction is almost 18%, 25% and 35% in SVC, SL+NC and SL systems respectively.

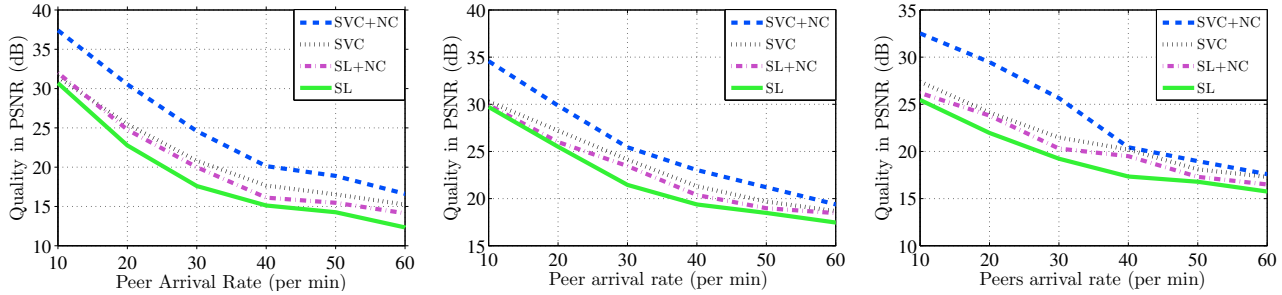
Impact of Churn Rate on Video Quality. We next study the impact of the churn rate on the streaming quality. In this scenario, we consider a highly dynamic peer-to-peer network with frequent arrivals and departures of peers. Maintaining a reasonable video quality in dynamic systems shows their robustness to frequent changes in network topology. In this experiment we will show that SVC+NC is more resilient and provides a more reliable peer-to-peer streaming system than the other systems.

In highly dynamic peer-to-peer systems, some peers join the system, start streaming and also contribute their resources to others. At the same time, other peers may be leaving the system, which will result in loss of upload resources and perhaps disruption of some on-going streaming sessions. We refer to the ratio of the total number of peers that join the streaming system during the simulation time to the total number of peers that leave the system as the churn rate. All arrivals and departures are scheduled according to a Poisson distribution during the simulation time. We vary the churn rate between 1 and 8. For each value of the churn rate. For example, a churn rate of 2 means that if x number of peers leave the system during the simulation time, $2x$ new peers will arrive during that period. A robust P2P stream-



(a) Sony Demo (b) Tokyo Olympics (c) NBC News

Figure 6: Impact of churn rate on the average video streaming quality.



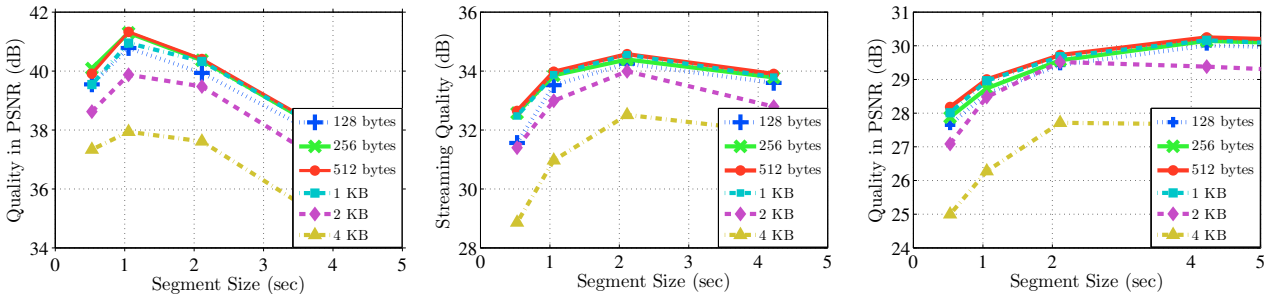
(a) Sony Demo (b) Tokyo Olympics (c) NBC News

Figure 7: Impact of flash crowd on video streaming quality.

ing system should utilize the resources brought by the new peers as well as provide them with good quality. We run the experiments for the three video streams in Table 1 and for each streaming system: SL, SL+NC, SVC, and SVC+NC. We measure the average quality perceived across all peers for each churn rate. We plot the results in Fig. 6. The results confirm the superior and stable performance of the proposed SVC+NC streaming system as several dBs in quality gain are observed in all cases. The figure also shows that as more peers join the quality for all peers improve, which is shown for high churn rates. This is because: the proposed SVC+NC system can: (i) increase the average throughput in the system since it uses network coding to harvest the resources of the new peers, and (ii) improve the quality by providing more video layers, which is enabled by the scalability nature of the video streams. We note that single layer streaming systems may actually suffer in presence of high churn rates, as shown in Fig. 6. This is because these systems take time to start effectively utilizing the resources of the new peers and they only provide a single version of the video streams. As the figure also shows, adding network coding to single layer streaming systems mitigates the first problem, but not the second: the average quality provided by SL+NC systems slightly improves as more peers join the system.

Impact of Flash Crowd Arrivals. In flash crowd arrivals, peers join the network in a short period of time. In this case, the demand for receiving the video data may become more than the available resources. Flash crowds scenarios put a substantial stress on the P2P streaming systems that strive to provide a reasonable and sustained video quality to peers. Addressing flash crowd arrivals is important for practical systems as they often happen after the release of popular video clips. We change the average number of peer arrivals per minute from 10 to 60 with an increment of 10. Peers arrive uniformly at random during the simulation period. We allow up to 10% of the active peers to leave during the streaming sessions, which also happen at uniform random times. We measure the average quality in dB for all considered systems for each peer arrival rate. The results, shown in Fig. 7, demonstrate that while under very high peer arrival rates the quality rendered by all systems decreases because of the limited upload capacity, the SVC+NC system provides relatively better quality in flash crowd scenarios than other systems. The figure shows that there is at least 2 dB quality difference by the SVC+NC and SL systems (right lower corner of Fig. 7(b)) and up to 7 dB (left top corner of Fig. 7(c)).

Impact of Segment and Block Sizes. Finally, we investigate the effect of the segment and block sizes on the streaming quality of the proposed SVC+NC system. We vary the segment size from 0.5 to 5 sec. For each segment size, we vary the block size from 128 bytes to 4 kilobytes and we run the experiments for each considered streaming system. We measure the average streaming quality and plot the results in Fig. 8. A few observations can be drawn from this figure. First, decreasing the block size for network coding (up to 512 bytes) generally yields better video quality. This is because when blocks are small, a single segment will have many blocks. This allows multiple sending peers to cooperate and send different (non-redundant) encoded blocks. On the other hand, decreasing the block size below 512 bytes



(a) Sony Demo (b) Tokyo Olympics (c) NBC News

Figure 8: Impact of Segment and Block Sizes on video streaming quality.

yields marginal or no additional benefits. The second observation is that, the ideal segment size (in sec) varies for different video streams. This is because the videos used in the experiments have diverse average bit rates of: 850, 500, 325 Kbps for the Sony Demo, Tokyo Olympics, and NBC News videos, respectively. From our experiments and the results shown in Fig. 8, we have found that a segment should contain about 100 to 200 KB of video data. Thus, the actual segment size (in sec) will depend on the bit rate of the video stream distributed to peers. For example, a segment size of 1 sec yields the best performance for the Sony Demo video according to Fig. 8(a). Given that the Sony Demo has an average bit rate 850 Kbps, the amount of video data in a segment is about 106 KB. Whereas a segment size of 4 sec provides the best performance for the NBC News video according to Fig. 8(c), which means that the segment will contain about $4 \times 325/8 = 162$ KB.

5. CONCLUSIONS AND FUTURE WORK

Most of the current P2P streaming systems use nonscalable video streams and thus they provide a single version for all receivers despite their diverse resources. These systems may also suffer from suboptimal utilization of peer upload bandwidth. In this paper, we showed that designing P2P streaming systems with scalable video coding and network coding can solve both of the above problems. We showed that the integration of the network coding *and* scalable video coding techniques improves the system performance beyond what is possible if each technique is used separately. We implemented the proposed system and conducted extensive evaluation study in realistic settings and with actual scalable video traces. The evaluation study confirms the significant potential performance gain, in terms of visual quality perceived by peers, average streaming rates, streaming capacity, and adaptation to high peer dynamics.

The work in this paper can be extended in multiple directions. For example, we are currently developing analytical models to analyze the performance of the proposed P2P live streaming system. We are also implementing the proposed system as a plug-in library that can be used in other streaming systems in order to enable them to benefit from scalable video streams and network coding methods.

6. REFERENCES

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

[2] I. Amonou, N. Cammas, S. Kervadec, and S. Pateux. Optimized rate-distortion extraction with quality layers in the scalable extension of H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1186–1193, September 2007.

[3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *Proc. of ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 298–313, Bolton Landing, NY, October 2003.

[4] X. Chenguang, X. Yinlong, Z. Cheng, W. Ruizhe, and W. Qingshan. On network coding based multirate video streaming in directed networks. In *Proc. of IEEE International Conference on Performance, Computing and Communications (IPCCC'07)*, pages 332–339, New Orleans, LA, April 2007.

[5] P. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. of Allerton Conference on Communication, Control, and Computing (Allerton'03)*, Monticello, IL, October 2003.

[6] Y. Cui and K. Nahrstedt. Layered peer-to-peer streaming. In *Proc. of ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'03)*, pages 162–171, Monterey, CA, June 2003.

[7] C. Feng and B. Li. On large-scale peer-to-peer streaming systems with network coding. In *Proc. of ACM Multimedia'08*, pages 269–278, Vancouver, Canada, October 2008.

[8] C. Fragouli, J. L. Boudec, and J. Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, January 2006.

[9] C. Gkantsidis, J. Miller, and P. Rodriguez. Anatomy of a p2p content distribution system with network coding. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS'06)*, Santa Barbara, CA, February 2006.

[10] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. of IEEE INFOCOM'05*, pages 2235–2245, Miami, FL, March 2005.

[11] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *Proc. of IEEE INFOCOM'06*, Barcelona, Spain, April 2006.

[12] M. Hefeeda and C. Hsu. Rate-distortion optimized streaming of fine-grained scalable video sequences. *ACM Transactions on Multimedia Computing, Communications and Applications*, 4(1):1–28, January 2008.

[13] O. Hillestad, A. Perkis, V. Genc, S. Murphy, and J. Murphy. Adaptive H.264/MPEG-4 SVC video over IEEE 802.16 broadband wireless networks. In *Proc. of IEEE Packet Video Workshop (PV'07)*, pages 26–35, Lausanne, Switzerland, November 2007.

[14] C. Huang, J. Li, and K. Ross. Can Internet video-on-demand be profitable? In *Proc. of ACM SIGCOMM'07*, pages 133–144, Kyoto, Japan, August 2007.

[15] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE Transactions on Information Theory*, 11(5):782–795, October 2003.

[16] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *Proc. of ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 282–297, Bolton Landing, NY, October 2003.

[17] X. Lan, N. Zheng, J. Xue, X. Wu, and B. Gao. A peer-to-peer architecture for efficient live scalable media streaming on Internet. In *Proc. of ACM Multimedia'07*, pages 783–786, Augsburg, Germany, September 2007.

[18] P. Larsson. Multicast multi-user ARQ. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'08)*, pages 1985–1990, Las Vegas, NV, April 2008.

[19] S. Li, R. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371, 2003.

[20] Z. Liu, Y. Shen, K. Ross, J. Panwar, , and Y. Wang. Substream trading: Towards an open P2P live streaming system. In *Proc. of IEEE Conference on Network Protocols (ICNP'08)*, pages 94–103, Orlando, FL, October 2008.

[21] N. Magharei and R. Rejaie. Prime: peer-to-peer receiver driven mesh-based streaming. In *Proc. of IEEE INFOCOM'07*, pages 1415–1423, Anchorage, AK, May 2007.

[22] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: a comparative study of live P2P streaming approaches. In *Proc. of IEEE INFOCOM'07*, pages 1424–1432, Anchorage, AK, May 2007.

[23] P. Maymounkov, N. Harvey, and D. Lun. Methods for efficient network coding. In *Proc. of Allerton Conference on Communication, Control, and Computing (Allerton'06)*, Urbana, IL, September 2006.

[24] K. Mokhtarian and M. Hefeeda. Efficient allocation of seed servers in peer-to-peer streaming systems with scalable videos. In *Proc. of IEEE International Workshop on Quality of Service (IWQoS'09)*, pages 1–9, Charleston, SC, July 2009.

[25] D. Nguyen, T. Nguyen, and B. Bose. Wireless broadcasting using network coding. In *Proc. of Workshop on Network Coding, Theory, and Applications (NetCod'07)*, pages 914–925, San Diego, CA, January 2007.

[26] K. Nguyen, T. Nguyen, and S. Cheung. Peer-to-peer streaming with hierarchical network coding. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'07)*, pages 396–399, Beijing, China, July 2007.

[27] J. Park, M. Gerla, D. Lun, Y. Yi, and M. Medard. Codecast: A network-coding-based ad hoc multicast protocol. *IEEE Wireless Communications*, 13(5):76–81, October 2006.

[28] D. Petrovic, K. Ramchandran, and J. Rabaey. Overcoming untuned radios in wireless networks with network coding. *IEEE Transactions on Information Theory*, 52(6):2649–2657, June 2006.

[29] PPLive. `http://www.pplive.com/`.

[30] R. Rejaie and A. Ortega. PALS: peer-to-peer adaptive layered streaming. In *Proc. of ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'03)*, pages 153–161, Monterey, CA, June 2003.

[31] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[32] SopCast. `http://www.sopcast.com/`.

[33] TVAnts. `http://www.tvants.com/`.

[34] UUSee. `http://www.uusee.com/`.

[35] Video Traces Research Group, 2009. `http://trace.eas.asu.edu/h264svc/`.

[36] M. Wang and B. Li. Lava: A reality check of network coding in peer-to-peer live streaming. In *Proc. of IEEE INFOCOM'07*, pages 1082–1090, Anchorage, AK, May 2007.

[37] 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, December 2007.

[38] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[39] Y. Wu, P. A. Chou, and K. Jain. A comparison of network coding and tree packing. In *Proc. of IEEE International Symposium on Information Theory (ISIT'04)*, Chicago, IL, July 2004.

[40] M. Zhang, L. Zhao, J. Tang, and S. Yang. A peer-to peer network for streaming multicast through the Internet. In *Proc. of ACM Multimedia'05*, Singapore, November 2005.

[41] X. Zhang, J. Liu, B. Li, and T. Yum. DONet/CoolStreaming: a data-driven overlay network for live media streaming. In *Proc. of IEEE INFOCOM'05*, pages 2102–2111, Miami, FL, March 2005.

[42] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang. Lion: Layered overlay multicast with network coding. *IEEE Transactions on Multimedia*, 8(5):1021–1032, October 2006.