A Network Coding Equivalent Content Distribution Scheme for Efficient Peer-to-Peer Interactive VoD Streaming

Yung-Cheng Kao, Chung-Nan Lee, Peng-Jung Wu, and Hui-Hsiang Kao

Abstract—Although random access operations are desirable for on-demand video streaming in peer-to-peer systems, they are difficult to efficiently achieve due to the asynchronous interactive behaviors of users and the dynamic nature of peers. In this paper, we propose a network coding equivalent content distribution (NCECD) scheme to efficiently handle interactive video-on-demand (VoD) operations in peer-to-peer systems. In NCECD, videos are divided into segments that are then further divided into blocks. These blocks are encoded into independent blocks that are distributed to different peers for local storage. With NCECD, a new client only needs to connect to a sufficient number of parent peers to be able to view the whole video and rarely needs to find new parents when performing random access operations. In most existing methods, a new client must search for parent peers containing specific segments; however, NCECD uses the properties of network coding to cache equivalent content in peers, so that one can pick any parent without additional searches. Experimental results show that the proposed scheme achieves low startup and jump searching delays and requires fewer server resources. In addition, we present the analysis of system parameters to achieve reasonable block loss rates for the proposed scheme.

Index Terms—Peer-to-peer, network coding, interactive operations, video-on-demand.

1 Introduction

Multimedia streaming is now a popular Internet service. However, efficient streaming to a large client population is hampered by server bandwidth constraints and the fact that IP-layer multicast is not universally supported. Peer-to-peer (P2P) collaborative streaming is a promising solution to the problem of efficiency. In a P2P system, each peer requests multimedia content from specific supplying peers. Then, after receiving the data, the peer caches it in local storage so that the (receiving) peer can now become a new supplier for other peers. An important challenge in a P2P collaborative video-on-demand (VoD) streaming system is to develop an effective content distribution scheme that can support a dynamic network among peers, where autonomic peers can join or leave the system at any time and any place in the network. The situation is further complicated by the need to support random access, such as the trick plays of pause/resume, jump, fast forward (FF), and rewind. Such trick plays may occur frequently. Most existing approaches require at least O(log(N)) time to locate the requested segment, where N is the number of segments of the requested video. The scheme we propose in this study can offer a more efficient approach (and one that supports trick plays) to the P2P-based interactive VoD systems.

The "cache-and-relay" technique used by Chang et al., Sharma et al., Cui et al., and Do et al. [18], [19], [20], [21], [22] keeps recently played data in the cache of the receiver so that it can be forwarded to other peers. BitTorrent [18] allows users to stream videos and watch them, even during download. This approach requires clients to cache the entire video file, even though they have already viewed it, thus wasting storage space. The cache-and-relay technique has difficulty with trick play operations. For instance, a parent peer might jump to another play point in the video. This would prevent it from forwarding a continuous stream to its child peers, thus requiring all its child peers to search for a new parent. As a result, it will cause propagation delay for the child peers.

The proposed scheme avoids these problems by adopting the additional static local *storage* used by Yiu et al. and Xu et al. [1], [2], instead of sliding window *playback buffering*, to efficiently support users' interactive operations and decrease complexity. The advantage of using additional storage is that any user interactivity on the part of the peer does not affect its children from continuing to receive its stored media data. Moreover, observations from a large number of user requesting logs [3] indicate that random seeking is frequently performed by most users. This is reasonable, as users usually jump directly to the scene of interest and skip boring segments. Therefore, it would be favorable if the system could guarantee peers the ability to jump to any play point in the requested video without searching for new parent peers that possess specific segments.

In this paper, we propose a novel network coding equivalent content distribution (NCECD) scheme for a multisource, P2P-based, interactive VoD system. So as to

- *Y.C. Kao and P.J. Wu are with the Department of Computer Science and Engineering, National Sun Yat-Sen University, No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan, R.O.C. and the Industrial Technology Research Institute, Hsinchu 31040, Taiwan, R.O.C. E-mail: m9034617@student.nsysu.edu.tw, wupl@cse.nsysu.edu.tw.*
- *C.-N. Lee and H.-H. Kao are with the Department of Computer Science and Engineering, National Sun Yat-Sen University, No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan, R.O.C. E-mail: cnlee@cse.nsysu.edu.tw, D953040006@student.nsysu.edu.tw.*

Manuscript received 24 Aug. 2010; revised 15 Feb. 2011; accepted 29 Aug. 2011; published online 30 Sept. 2011.
Recommended for acceptance by D. Epema.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-08-0497.
Digital Object Identifier no. 10.1109/TPDS.2011.244.

1) enable child peers to link to these parents with partial—and not duplicated—data for the complete video and 2) tackle the problem of parent departure, we use linear network coding to generate an encoded block by encoding all blocks in one segment. If enough encoded blocks are received by a child peer, the child peer can decode the original segment. Therefore, linear network coding, combined with interleaving block distribution, results in a situation in which a child peer only needs to find a sufficient number of parent peers to be able to view any given segment of the requested video; the child peer does not have to search for new parent peers to view the next segment or to perform interactive operations (e.g., jump or rewind). Furthermore, when a parent peer leaves the network, the child peer can still receive some encoded blocks from other parents so as to decode the original segment. Besides, the child peer can locate any peer that caches encoded blocks of the requested video as a parent peer in the P2P networks, thereby obtaining good video segment search performance.

The main contributions of this study are as follows:

1. Using interleaving block distribution and linear network coding techniques, we propose an efficient media data distribution scheme for P2P-based, interactive VoD streaming.
2. The proposed content distribution scheme is cost effective as it does not need to maintain an index data structure and network topology to locate parent peers that cache target segments.
3. P2P-based trick plays are supported effectively and naturally.
4. The study analyzes practical system parameters such as block size, peer cache capacity, required number of parents, linear network coding implementation, and packet loss rates from parent peer departure.

The rest of this paper is organized as follows: Related work is introduced in Section 2. Section 3 describes the proposed NCECD scheme. Section 4 presents a performance evaluation of the proposed scheme and competing schemes. Finally, in Section 5, conclusions are drawn.

2 Related Work

Some P2P systems using linear network coding have already been developed. Wang and Li [7] presented R^2, a new streaming algorithm designed to combine random network coding with a randomized push algorithm. R^2 focuses on improving the efficiency of live streaming in terms of startup buffering delays, resilience to the unpredictable behavior of peers, and bandwidth saving of streaming servers. Wu et al. [10] proposed some decentralized strategies to eliminate conflicts among coexisting streaming overlays on contested bandwidth and combined those strategies with network-coding-based content distribution to realize efficient multi-overlay streaming. Chi et al. [11] proposed a buffer-assisted search (BAS) scheme to increase partner search performance by decreasing the size of the index structure. They also designed a novel scheduling algorithm using the dead-line aware network coding (DNC) to fully utilize network resources by adopting an appropriate coding window size. Gkantsidis et al. [12] implemented a P2P content distribution system that uses network coding. They gave a detailed performance analysis of one such P2P system to show that network coding is practical as it generates little overhead, both in terms of CPU processing and I/O activity. Feng and Li [16] analyzed and compared the performance of network-coding-based P2P streaming to that of traditional pull-based P2P streaming. Annapureddy et al. [13] used network coding for P2P VoD services by efficiently distributing and scheduling video segments, thereby achieving high system performance. Nguyen et al. [14] used a network coding technique to reduce duplicated storage and remove the requirement of tight synchronization between senders. Gkantsidis and Rodriguez [23] proposed a new scheme for the content distribution of large files using network coding. They designed and implemented a system, called Avalanche, and explored various practical issues in network coding. A major issue in any content distribution scheme is protection against malicious peers. Avalanche uses special sets of secure hash functions that support network coding operations; further, it requires very few computational resources. Kim et al. [24] quantified the impact of Byzantine attacks on the coded system by estimating the probability that a receiver node cannot recover a file successfully. They proposed a novel signature scheme that performs packet-level Byzantine detection. Chang et al. [19] applied a linear programming approach with network coding to the evaluation of download finish times in a P2P network. They disproved the hypothesis that Min-Min scheduling yields the minimum average finish time for routing, and showed that coding can provide a robust optimal solution and better performance than routing in a dynamic network environment. Some works [25], [26] have considered the implementation of network coding and p2p cooperative computing for wireless network and mobile devices. Dimakis et al. [27] have introduced the notion of regenerating codes, which make a new peer only need *functions* of the stored data from the surviving parent peers to significantly decrease the repair bandwidth. Liu et al. [28] have implemented a P2P storage cloud for on-demand streaming called *Novasky*. A coding-aware peer storage and replacement strategy as well as an adaptive server push strategy are proposed to achieve storage efficiency, durable video availability, load balancing, and the balance of the demand and supply of data and bandwidth in the P2P storage cloud. However, previous works have not considered the following three issues. First, previous studies have not yet found a way in which to divide and distribute the data from one entire video to one parent peer or sender to allow any parent peer to easily replace leaving parent peers. Second, previous studies do not consider that an appropriate number of parent peers for multisource streaming by the network coding technique must be determined to reduce the block loss rate from parent peers' departure and the dependent encoded blocks. Third, the impact of interactive activities on a P2P VoD streaming system has also not been considered. These three issues are analyzed and resolved in this paper.

3 The Proposed Scheme

This section first introduces the interleaving distribution approach to the distribution of blocks of segments to peers

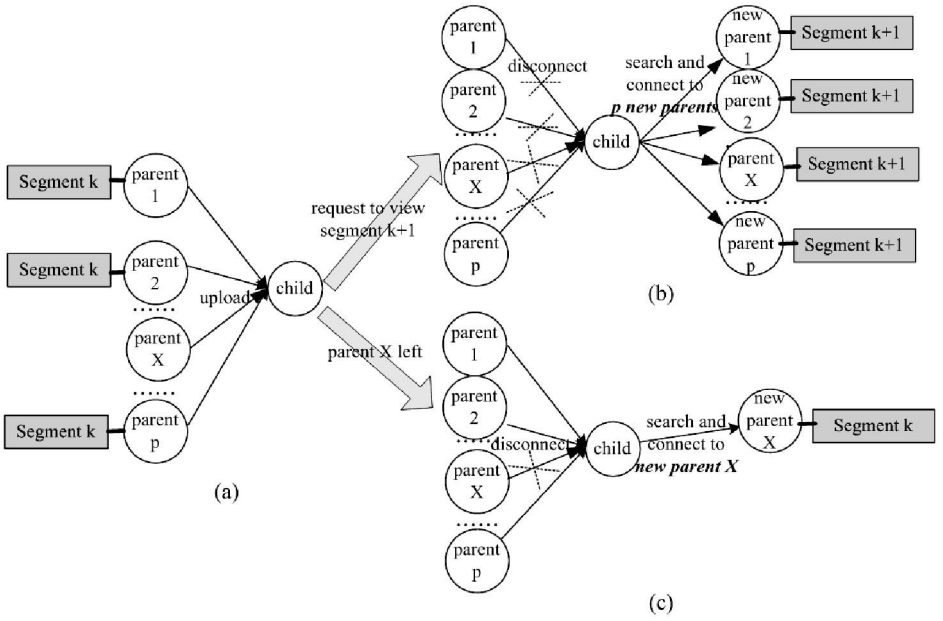


Fig. 1. The basic cache-and-relay approach with multiple parents. (a) Each parent peer caches a piece of a video, or one complete segment. (b) When the child peer wants to view the next segment $k+1$ (or jump to another segment), it must first disconnect from the original p parent peers and locate p new parent peers for the service or (c) when a parent peer leaves, the child peer needs to find a new parent peer that also stores the same segment k.

to avoid having to locate new parent peers when the child peer requests a new segment. Then, a network-coding approach is presented to simplify the search for a new parent peer when a parent departs. Next, we describe how to apply the proposed scheme to the P2P VoD system. An analysis of practical system parameters is given in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.244.

3.1 Interleaving Distribution of Segments

While explaining the details and benefits of the NCECD scheme, it is helpful to provide some comparisons with existing approaches. We first consider the basic cache-and-relay technique with multiple parents, as described by Yiu et al. and Xu et al. [1], [2]; each peer caches one complete video segment. In this way, when a child peer wants to view a segment, it first locates the peers that have cached the requested segment and then makes a request to these peers (for downloading and playing this segment). Fig. 1 depicts the method and shows how it handles new segment searches or lost peers. When the child requests a new segment, it has to search for p new parents that have this new segment. The child peer still needs to pay the search cost which is bounded by O($\log P$) in VMesh, O($\log N$) in BBTU, and O($\log(P/\log P)$) in DSL, where P is the number of peers that view the video, and N is the number of segments of the video.

As a first step toward the proposed NCECD scheme, consider the proposed basic interleaving scheme where each peer caches interleaving data from all segments of the video, as shown in Fig. 2. Parent peer 1 caches the 1st block of all N segments, parent peer 2 caches the 2nd block of all N segments, ..., parent peer M caches the Mth block of all N segments. When a newly arrived child connects to M parents that each caches N blocks, they, together, cache all M^*N blocks for N segments of the video. The child peer would not

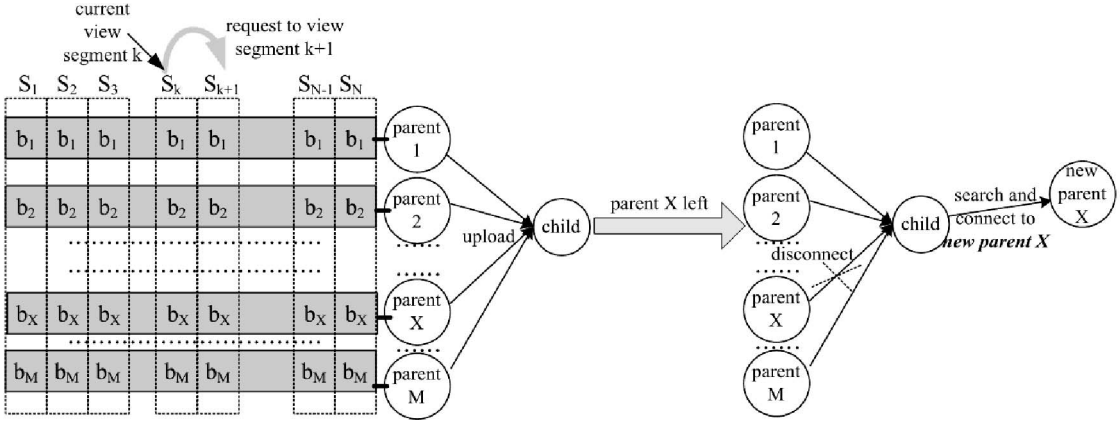


Fig. 2. The proposed basic interleaving scheme.

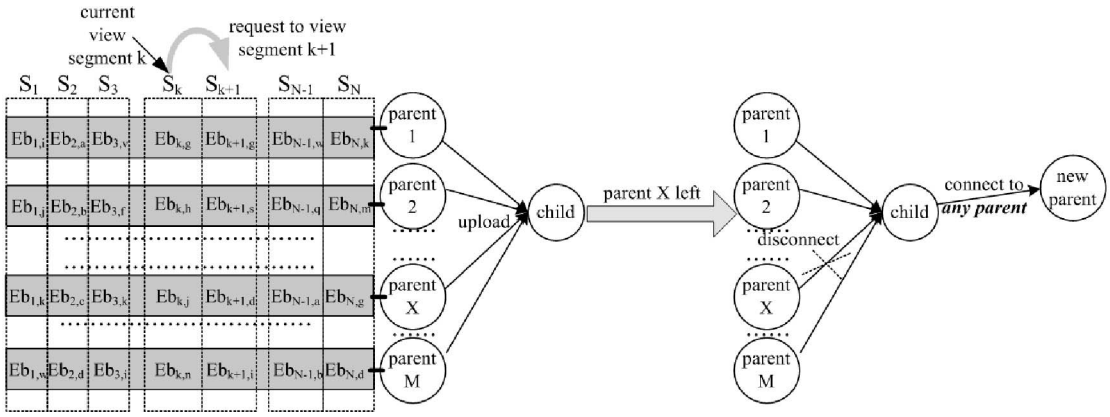


Fig. 3. The proposed NCECD scheme.

need to search for new parents for the whole video length, even when requesting the next segment or performing jump operations. Although the jump delay is reduced to zero, the startup time is high because the child must locate parents that cache all different blocks. Furthermore, whenever a parent holding block X of all segments leaves, the child peer pays the cost of locating a new parent that has the same block X. This motivated us to integrate linear network coding into the proposed basic interleaving scheme to further reduce the startup time and search cost. In Section 3.2, we will explain how NCECD uses a novel content distribution strategy to avoid the high new parent searching cost.

3.2 Linear Network Coding of Segments

To reduce the cost of finding parents, the proposed NCECD scheme, as illustrated in Fig. 3, utilizes linear network coding technology [6] for media data distribution. When a sufficient number of encoded blocks are read from M different parents, the original segment can be decoded. The proposed NCECD scheme is a significant advance over the approach shown in Fig. 2 because it has a much faster parent search time. Whereas the basic interleaving scheme requires the child to find a specific parent with the desired block, the NCECD scheme does not require the search for specific parents.

Fig. 4 shows more a detailed illustration of the proposed scheme: the original data blocks of segment X, for $X_{\in(1,2,\ldots N)}$, are denoted by $[b_{X,1}, b_{X,2}, b_{X,3}, \ldots, b_{X,M}]$. An encoded block $Eb_{X,i}$ is generated by the combination of encoding $[b_{X,1}, b_{X,2}, b_{X,3}, \ldots, b_{X,M}]$ of segment X using an independent and random set of coding coefficient vectors $f_i = [c_{i,1}, c_{i,2}, \ldots, c_{i,M}]$ in the Galois field $\mathrm{GF}(2^8)$, one for one block in the segment. An encoded block $Eb_{X,i}$ is given by $\sum_{j=1}^{M} c_{i,j} \cdot b_{X,j}$.

The value of the encoding coefficient vector f_i must be transmitted from the parent peer to the child peer, and thus, an overhead of M bytes per encoding block is needed. A peer only receiving any M linear independent encoded blocks $E = [Eb_{X,1}, Eb_{X,2}, \ldots, Eb_{X,M}]$ can thus recover the original segment X. First, the coding coefficient vector $f_i = [c_{i,1}, c_{i,2}, \ldots, c_{i,M}]$ of each encoded block $Eb_{X,i}$ is used to form an $M \times M$ matrix F. Each row of F is composed of coding coefficients of one encoded block $Eb_{X,i}$. The original blocks of segment $X = [b_{X,1}, b_{X,2}, b_{X,3}, \ldots, b_{X,M}]$ are then recovered as follows:

$$X = \mathrm{F}^{-1}\mathrm{E}^{\mathrm{T}}.$$

In this way, each parent peer caches N encoded blocks separately from N segments of the video. As shown in Fig. 3, each parent peer caches an encoded block of segment 1, an encoded block of segment $2, \ldots,$ and an encoded block of segment N. Therefore, the child peer only needs to connect to M parent peers to receive M linear independent encoded blocks of any segment and thus is able to recover any particular required segment.

Note that in Fig. 3, the content cached by parents 1 to M are equivalent, but not the same; in most cases, an arbitrary peer could replace any of the parents even though their data are different. With encoded blocks, whenever a child peer tries to find a new parent, almost any parent peer with sufficient available bandwidth on the system can be selected. This, of course, results in a very low search cost.

Moreover, the proposed scheme can naturally support fast-forward and fast-rewind (FR) operations. If the video is played in FF, then the video needs to be downloaded faster, necessitating a larger bandwidth. For some existing systems, the child peer has to switch its parent peers more frequently when performing FF or FR operation. For example, if a peer plays a 6-min segment at 4× speed, it must switch to new parents after 1.5 min. This problem would be naturally solved by the proposed scheme: a child peer does not need to switch parent peers at any point when watching the video. In addition, since any parent peer can be selected, a child peer can easily increase the number of parent peers to aggregate the bandwidth required to support FF and FR operations.

3.3 Application of the NCECD Scheme to P2P VoD Streaming

In the proposed NCECD scheme, in addition to streaming the media data of the desired segments, the participant peers also download other encoded blocks from parent peers or the server until they have finished downloading N encoded blocks separately from N segments. At the same time, participant peers may directly forward these downloaded blocks to other peers or reencode a received segment with a random coefficient vector and then forward these reencoded blocks to maintain high independence among distributed encoded blocks in the P2P network. An adaptive incentive mechanism proposed by Habib and Chuang [17]

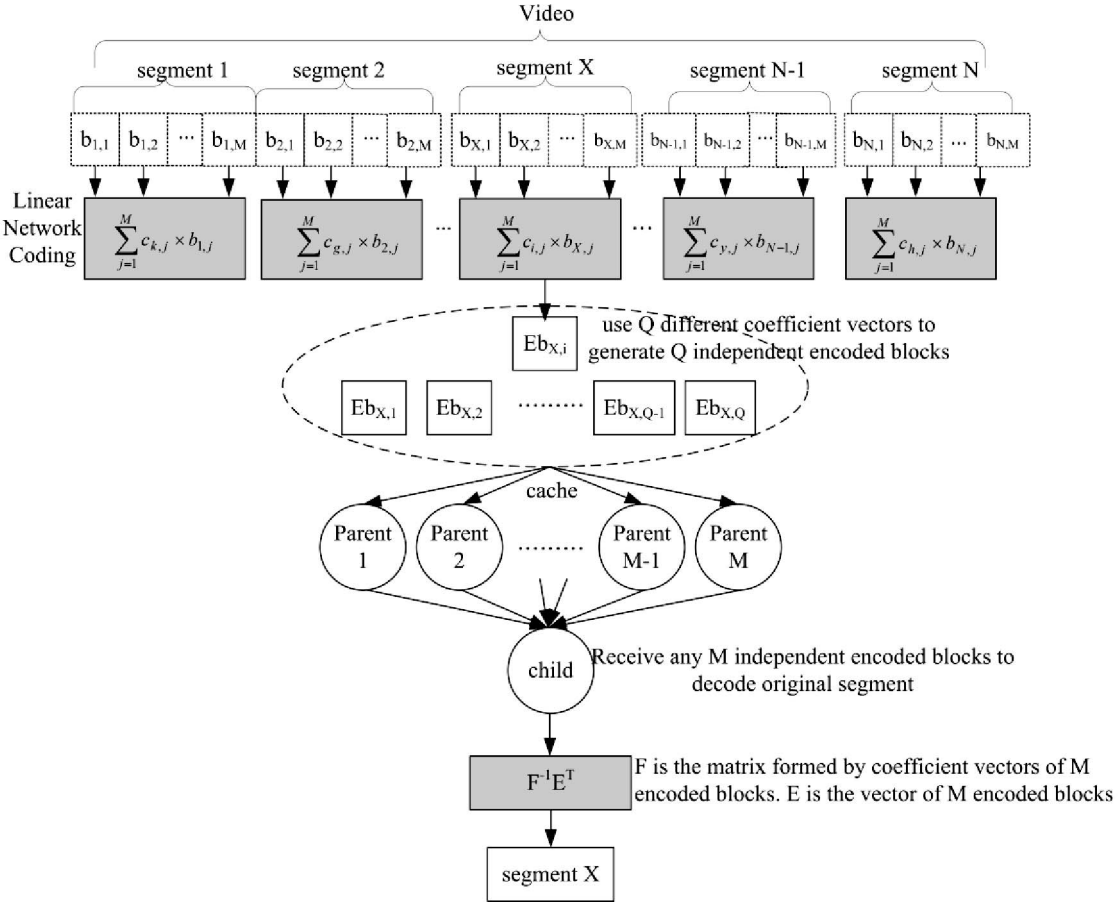


Fig. 4. The network coding technique is used to encode video blocks. One video is divided into N segments and each segment is divided into M blocks. Via network coding, M blocks of a segment become an encoded block. In this way, different encoding coefficients are used to generate Q independent encoded blocks. Each parent peer caches one encoded block; one child peer connects to M parent peers to receive M independent encoded blocks to decode original segment X.

is required to encourage peers to use these cooperative behaviors; however, this topic lies outside the scope of the present paper.

Next, we describe the process by which a new peer joins the system. Since we focus on exploring the impact of network coding on P2P VoD streaming, we use a simplified and more general joining scheme to explain how a new peer finds and connects to a sufficient number of parent peers. When the arriving peer wants to join the system, it first contacts the centralized server by sending a joining message so as to receive a list of active peers on the system. If the number of active peers on the system is not able to form a candidate list, the arriving peer connects to the server directly to stream all video segments. Otherwise, the new arriving peer selects $K(> M)$ peers from the candidate list as its parent peers (e.g., in terms of sufficient uploading bandwidth and the shortest RTT), then makes requests to these parent peers for downloading of the encoded video blocks. If each parent peer can provide N encoded blocks, the newly arriving peer only needs to find M parent peers to be able to view the whole video ($N \times M$ blocks). Thus, the arriving peer must check whether there are at least M parent peers out of K connecting parent peers that can provide a total of $N \times M$ encoded blocks. If this condition is not met, the arriving peer has to connect to additional peers until this requirement is met. These additional connecting ($K - M$) peers would transmit additional network coding packets to the child peer either to recover lost blocks from parent peers' departures or to compensate dependent encoded blocks. These ($K - M$) peers might not all cache N encoded blocks of N segments; they only provide what has been cached. Thus, for a receiver at this stage, this scheme gives different recovery capabilities for different segments. Until these ($K - M$) peers finish downloading and caching all N encoded blocks, they can give the same (that is, most) recovery capabilities for each segment.

When the new peer has connected to a sufficient number of parent peers, and once enough data have been received to fill its buffer, the peer starts to play the video. The new peer simultaneously uses its remaining bandwidth to download and store other encoded video blocks from other peers (or possibly from the server) into its local cache. The cached content would then be used to serve other peers. Note that it would be better for a new peer to choose to download encoded blocks from other remote parent peers, instead of the local parent peers, so as to store encoded blocks that are independent of those cached by local peers. In this way, parent peers in the same local network have a higher probability of holding linearly independent encoded blocks. If the new peer elects to download encoded blocks from the server, the server would choose coefficient vectors in a random manner to generate new encoded blocks. In

this way, a child peer that connects to parent peers in the local network is very likely to receive enough independent encoded blocks.

Some typical systems, such as VMesh [1], BBTU [2], and Dynamic Skip List (DSL) [4], have been developed to support efficient, interactive VoD services (e.g., random seeking) in P2P networks. In VMesh [1], each peer maintains a list of peers storing either the previous or the next video segments. According to the list, peers can locate other peers with the subsequent required segments. In BBTU [2], a balanced binary tree is built according to BATON [5]. Each peer stores "links" to its parent peers, child peers, and in-order traversal peers. Video segments are stored (based on playback time) in peers' pre-fetching buffers along with in-order traversal of the tree. The search cost in BBTU is bounded by $O(\log N)$, where N is the number of segments of one video. Additionally, in DSL [4], peers are organized into layers. A fast skipping operation is introduced, which rapidly finds peers with the expected segments based on logical links to its neighbors in each layer. Both the search cost and the maintainability of the state information at a peer are constant or logarithmic with DSL size. In addition, a linear network-coding-based algorithm is developed to enable efficient data downloading from multiple sources in the DSL system. However, DSL does not utilize linear network-coding features to distribute video data among peers so as to reduce the cost of locating the required video segments.

In summary, in the proposed scheme, it is not necessary to maintain a costly specific network topology such as mesh, balanced tree, or dynamic skip list for locating parent peers to download target segments. It is only necessary that a peer connects to sufficient (arbitrarily chosen) parent peers to be able to view the whole video instead of maintaining an extra network topology. The proposed scheme can be integrated with any kind of codec since it is performed on the compressed video data to provide additional protection from packet loss and improve the performance of trick play. In terms of multiresolution codec, it is necessary to consider some additional factors such as the fact that high-bandwidth peers may receive more video layers, and thus, can contribute more video layers to peers, and the fact that more important layers should be cached by more peers.

4 Performance Evaluation

In Section 4.1, we describe the system model used in the simulation. The metrics used to measure the performance are also introduced. Finally, simulation results are presented in Section 4.2.

The proposed NCECD scheme is compared to some existing approaches, such as VMesh [1], BBTU [2], and DSL [4]. However, the DSL system adopts the cache-and-relay approach; therefore, we modify the DSL system to allow the peers to prefetch data into the extra allocated buffer based on the scheme used by the DSL system.

4.1 Simulation Model

In the following simulations, the length and bit rate of the video are 120 min and 480 kbps, respectively. Each peer is allocated an extra local buffer to cache 6 min of video, approximately 21 Mbytes. Thus, using the NCECD scheme, each child peer must connect to at least $120/6 = 20$ parent peers to view the whole video. When a new peer arrives, it requests the video from the beginning. In delay-related experiments, we use NS2 to evaluate the performance. The transit-stub model [9], which has a 2-level hierarchical topology, is used in the simulations. The GT-ITM [9] is utilized to generate the core network of 1,100 routers, where there is one transit domain with, on average, 11 routers. Each transit domain router has 11 stub domains with, on average, nine routers. A video server and 1,000 peers are randomly attached to stub-domain routers through a 100 Mbps duplex-link. To simulate the effect of cross traffic, each peer builds an additional UDP traffic flow connection to one of the other peers in the simulation. The traffic flow follows an on-off exponential distribution that is assigned a burst time of 500 ms and an idle time of 500 ms. The packet size is 1,024 bytes and the transmission rate is 200 kbps.

The following performance metrics are used to evaluate the proposed scheme. 1) **Startup delay**—the duration between the time that a client arrives to that when the client connects to the parent peers required to begin viewing the video. This metric focuses on the delay in finding the required parent peers at the startup stage, and the delay of buffering several seconds (e.g., one data segment) of media data. In addition, in the proposed scheme, the delay must also include the decoding time for the first received network coding segment. 2) **Jump delay**—the duration between the time that a client requests a jump operation to the time that the client connects to the required parent peers (and begins viewing the video). This metric focuses on the delay in finding the required parent peers at the jump stage, and thus, the delay of buffering several seconds (e.g., one data segment) of media data. Similar to startup delay estimation, the delay must also include the decoding time for the first received network coding segment in the jump operation. A lower jump delay indicates that the system can more efficiently support jump operations. 3) **The relationship between segments supplied to and demanded by peers**—the difference needs to be compensated for by the media server to support the entire system. Less difference means that the server stress is lower, and thus, that the system is more stable and scalable.

4.2 Simulation Results

1) Startup and jump delays. Startup delay indicates the first requested segment location latency. For an accurate comparison of segment location latency, a child peer must always try to first connect to the closest parent peer in all schemes. For the competing schemes and the proposed scheme, a child peer contacts a number of parent peers and begins downloading media data in parallel. For the competing schemes, a child peer only needs to find the first parent peer to start streaming. While the child peer downloads, it continues to connect to other parent peers for multisource streaming. In the proposed scheme, a child peer must contact enough parent peers to start streaming due to the network coding limitation. However, since all parent peers can be selected, the proposed NCECD scheme has a higher probability of connecting to closer parent peers for downloading, and it does not need to perform a costly search for parent peers through a segment distribution topology, as used by other competing schemes. Additionally, in the proposed scheme, a child peer can simultaneously send requests to multiple

TABLE 1
Startup Searching Time Complexity

Scheme	Startup Searching Time Complexity
NCECD	O(1)
VMesh	O(log$P$)
BBTU	O(log$N$)
DSL	O(log(P/log$P$))

TABLE 2
Jump Searching Time Complexity

Scheme	Jump Searching Time Complexity
NCECD	0
VMesh	Min(O(log$P$), linked hops)
BBTU	O(log$N$)
DSL	1+O(log(P/log$P$))

parent peers; as a result, the required delay is limited to the slowest connection time to the specific parent peer, as opposed to the accumulated time of all connection times to all parent peers.

First, we list the startup and jump searching time, which only includes the delay for searching target parent peers; we do not consider buffering or decoding time. Table 1 lists the startup searching time complexity. VMesh performs the worst because it uses a DHT search for locating the segment of interest at the startup stage. The DHT searching complexity is bounded by log(P), where P is the number of peers viewing the video. Clearly, the NCECD scheme performs better than the other competing schemes. Similarly, jump searching delay creates the seeking segment location latency during playback. Table 2 lists the jump searching time complexity. The trend is similar to that of the startup searching delay for other competing schemes. In VMesh, when a peer jumps to another segment, the scheme would choose to use a DHT search or linked lists to locate the segment of interest based on the required contacting hops; this is because peers maintain linked lists to the parent peers that have cached the previous or the next segment. In the DSL scheme, a jump operation is performed by leaving and rejoining the system, and thus, the jump searching time complexity is $1 + \mathrm{O}(\log(P/\log P))$. Note that the proposed NCECD scheme does not generate any searching delay when a child peer requests a jump operation. This is because in the proposed scheme, a child peer need only connect to sufficient parent peers to view the whole video; thus, jump searching has a delay of zero.

Next, we will consider buffering time and decoding time in order to estimate the complete startup and jump delays in all schemes. So as to reduce the decoding delay from decoding network coding blocks, we adopt the Gauss-Jordan elimination implemented in the decoding process [15]. As described by Wang and Li [15], using progressive decoding, the decoding time is almost completely covered by the buffering time, and thus, is almost negligible.

Fig. 5a shows the startup delay including searching, buffering, and decoding delay (only for the proposed scheme used to decode network coding blocks to receive the original segment) for a variety of average peer populations. The buffering delay refers to the time required to download the data of one segment for buffering. As the

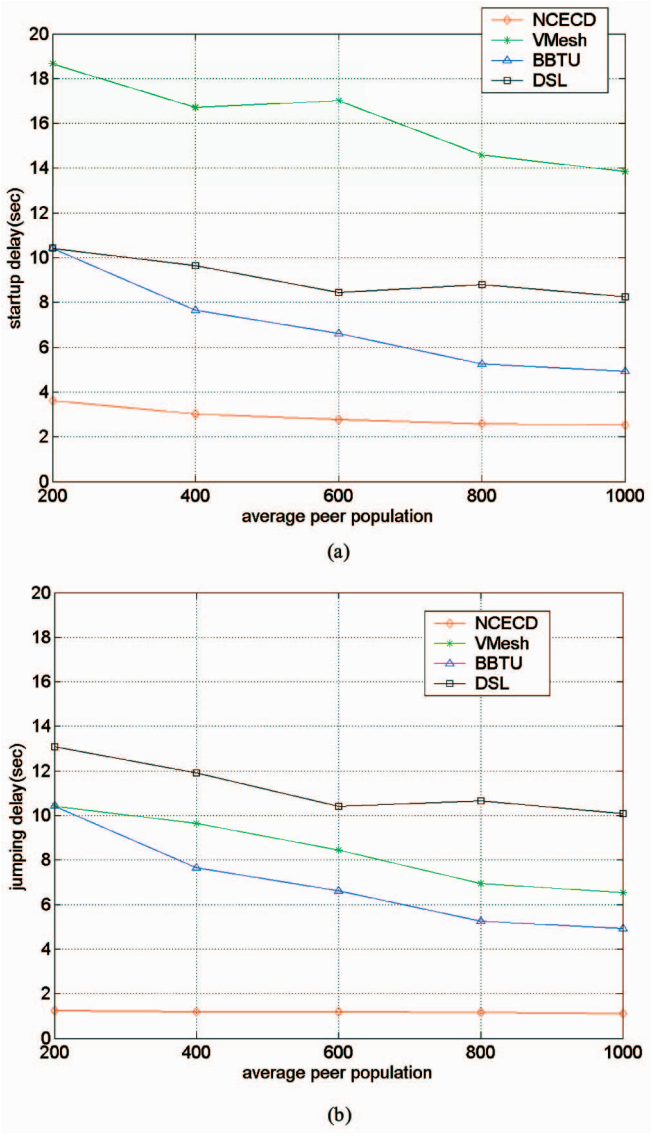


Fig. 5. (a) Startup delay under different average peer populations. (b) Jump delay under different average peer populations.

peer population increases, it is most likely that child peers in systems locate closer parent peers, thus reducing the average segment searching delay. VMesh performs the worst since it uses a DHT search to locate the segment of interest at the startup stage. Note that for VMesh, as the peer population increases, the startup searching delay should also increase. However, a child peer is also more likely to find closer parent peers, and therefore, decrease the startup searching delay. Thus, in VMesh, the startup delay is reduced when the peer population increases. The NCECD scheme performs better than BBTU, DSL, and VMesh, by, on average, 56.7, 68.3, and 82 percent, respectively. Similarly, jump delay incorporates the seeking segment location latency, buffering delay, and decoding delay (in the proposed scheme). Fig. 5b plots the jump delay under various average peer populations. The trend is similar to that of the startup delay observed in other competing schemes. Note that the proposed NCECD scheme does not generate searching delay, only buffering and decoding delays, when a child peer requests a jump operation. This is because a child peer only needs to connect to a sufficient number of parent peers to view the whole video; thus, the jump searching delay is zero.

2) The relationship between segments supplied to and demanded by peers: To evaluate the required server resources and bandwidth for this kind of P2P system, we first explore the relationship between supply and demand among peers. When supply and demand are balanced, most requests of child peers can be served by parent peers, making server stress low. Fig. 6a shows the duplication number of each segment distributed to all peers in a system. The segment popularity model from [1] is indicated by "popularity model" in Fig. 6a. For a balanced relationship of supply and demand, the segment duplication number should be proportional to its popularity. Therefore, if the duplication number of one distribution scheme can more closely match the "popularity model," it suggests that the distribution scheme can achieve a more balanced relationship between supplying peers and segments demanded by other peers. As shown in Fig. 6a, each parent peer in the proposed NCECD scheme is equivalent, in other words, one parent peer can be replaced by any other parent peer. Since one peer is a demander and also a supplier, if cached content of each peer is equivalent, it implies that the relationship of supply and demand is balanced. Hence, the duplication number most closely matches the "popularity model" for the proposed NCECD scheme. In some existing systems, to reduce system complexity, a peer downloads one random segment for caching; thus, the number of duplications of each segment in a video is almost the same in such a system. The uniform distribution scheme cannot completely balance the relationship between supply and demand. In VMesh, the popularity-aware distribution scheme is proposed to ascertain the popularity of each segment. The popularity of each segment is estimated by exchanging information between neighbor peers. The more the exchange of information, the more precise will be the estimated popularities of the segments. As shown in Fig. 6a, the lines indicated by 20, 100, and 1,000 peers present the duplication number of each segment when the peer number for exchanging information is 20, 100, and 1,000, respectively. As the amount of information being exchanged increases, the result gets closer to the "popularity mode;" however, it is costly for a number of peers to periodically exchange and update popularity information for segments. Fig. 6b plots the difference of requested segments by child peers and supplied segments by parent peers. The server needs to handle requests for segments whose demand exceeds supply. Thus, the larger the gap, the higher is the server stress. The proposed NCECD scheme outperforms the uniform and the popularity-aware schemes at 20, 100, and 1,000 peers by 90.3, 88.2, 78.9, and 43.9 percent, respectively.

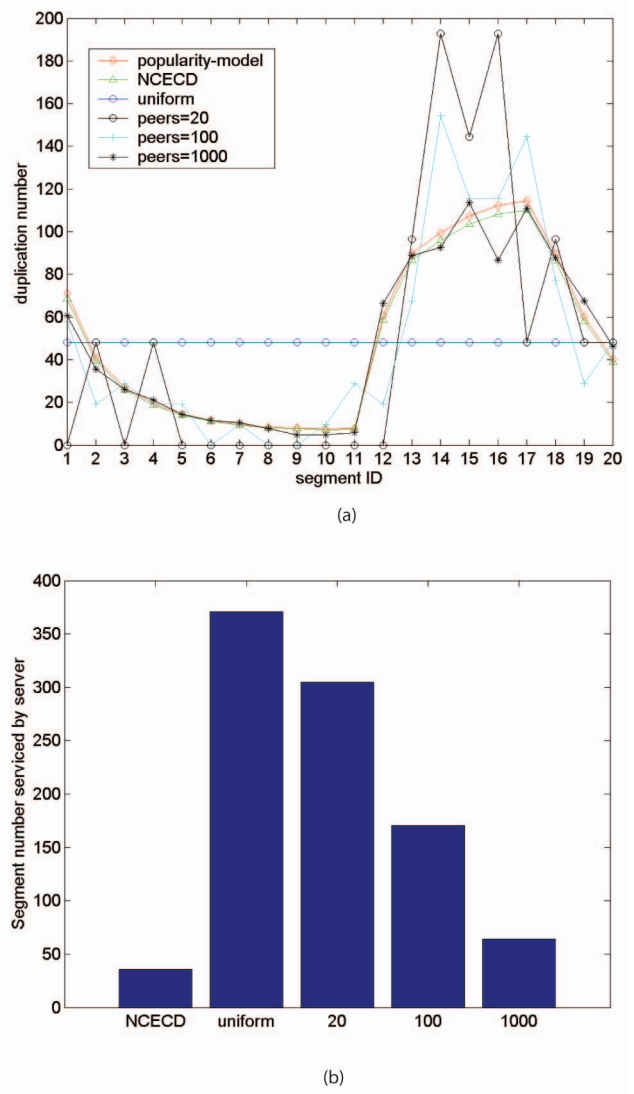


Fig. 6. (a)The duplication number of segments under different schemes. (b) The number of segments serviced by the server.

5 Conclusion

In this paper, we proposed a novel data distribution scheme called NCECD to provide interactive VoD services in a P2P network. In the NCECD scheme, videos are divided into smaller segments, which are further divided into blocks. The NCECD scheme applies network coding technology to generate several encoded blocks by combining the encoding of all blocks in one segment. These encoded blocks are distributed to peers on the system. A child peer needs only to find and link to a sufficient number of parent peers to view the entire video, thus eliminates the search for new parent

peers. In this way, interactive functionality can be supported efficiently. Network coding techniques naturally provide failure-tolerant streaming services as a client on the system connecting to multiple parent peers who have stored equivalent media data and are able to stream media data in parallel and collaboratively. An appropriate number of extra parent peers is found to provide low block loss probability.

Experimental results show that the NCECD scheme substantially relieves server stress by optimally matching supply of and demand for segments in a P2P network. Simulation and analyses demonstrate that the proposed scheme outperforms other competing schemes such as VMesh, BBTU, and DSL in terms of startup delay, jumping delay, and server stress. Additionally, NCECD can achieve very low block loss probabilities under various system parameters by connecting to an appropriate number of extra parent peers, allow for failure-tolerant streaming services in a P2P network.

REFERENCES

[1] W.P.K. Yiu, X. Jin, and S.H.G. Chan, "VMesh: Distributed Segment Storage for Peer-to-Peer Interactive Video Streaming," *IEEE J. Selected Areas in Comm.,* vol. 25, no. 9, pp. 1717-1731, Dec. 2007.

[2] C. Xu, G.M. Muntean, E. Fallon, and A. Hanley, "A Balanced Tree-Based Strategy for Unstructured Media Distribution in P2P Networks," *Proc. IEEE Int'l Conf. Comm. (ICC '08),* pp. 1797-1801, May 2008.

[3] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," *Proc. ACM Workshop Advances in Peer-to-Peer Multimedia Streaming,* pp. 29-38, Nov. 2005.

[4] D. Wang and J. Liu, "A Dynamic Skip List-Based Overlay for On-Demand Media Streaming with VCR Interactions," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 4, pp. 503-514, Apr. 2008.

[5] H.V. Jagadish, B.C. Ooi, and Q.H. Vu, "BATON: A Balanced Tree Structure for Peer-to-Peer Networks," *Proc. Int'l Conf. Very Large Data Bases (VLDB '05),* pp. 661-672, Aug. 2005.

[6] S.Y.R. Li, R.W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. Information Theory,* vol. 49, no. 2, pp. 371-381, Feb. 2003.

[7] M. Wang and B. Li, "R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming," *IEEE J. Selected Areas in Comm.,* vol. 25, no. 9, pp. 1655-1666, Dec. 2007.

[8] P.J. Wu, J.N. Hwang, C.N. Lee, C.C. Gau, and H.H. Kao, "Eliminating Packet Loss Accumulation in Peer-to-Peer Streaming Systems," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 19, no. 12, pp. 1766-1780, Dec. 2009.

[9] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE INFOCOM,* vol. 2, pp. 594-602, Mar. 1996.

[10] G. Wu, B. Li, and Z. Li, "Dynamic Bandwidth Auctions in Multioverlay P2P Streaming with Network Coding," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 6, pp. 806-820, June 2008.

[11] H. Chi, Q. Zhang, J. Jia, and X. Shen, "Efficient Search and Scheduling in P2P-Based Media-on-Demand Streaming Service," *IEEE J. Selected Areas in Comm.,* vol. 25, no. 2, pp. 119-130, Jan. 2007.

[12] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive View of a Live Network Coding P2P System," *Proc. ACM SIGCOMM Conf. Internet Measurement (IMC '06),* pp. 177-188, Oct. 2006.

[13] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez, "Exploring Vod in P2P Swarming Systems," *Proc. IEEE INFOCOM '07,* pp. 2571-2575, May 2007.

[14] K. Nguyen, T. Nguyen, and S.-C. Cheung, "Peer-to-Peer Streaming with Hierarchical Network Coding," *Proc. IEEE Int'l Conf. Multimedia and Expo,* pp. 396-399, July 2007.

[15] M. Wang and B. Li, "Lava: A Reality Check of Network Coding in Peer-to-Peer Live Streaming," *Proc. IEEE INFOCOM,* pp. 1082-1090, May 2007.

[16] C. Feng and B. Li, "On Large-Scale Peer-to-Peer Streaming Systems with Network Coding," *Proc. ACM Int'l Conf. Multimedia,* pp. 269-278, Oct. 2008.

[17] A. Habib and J. Chuang, "Service Differentiated Peer Selection: An Incentive Mechanism for Peer-to-Peer Media Streaming," *IEEE Trans. Multimedia,* vol. 8, no. 3, pp. 610-621, June 2006.

[18] http://www.bittorrent.com, 2011.

[19] C.S. Chang, T. Ho, M. Effros, M. Medard, and B. Leong, "Issues in Peer-to-Peer Networking: A Coding Optimization Approach," *Proc. IEEE Int'l Symp. Network Coding,* June 2010.

[20] A. Sharma, A. Bestavros, and I. Matta, "dPAM: A Distributed Prefetching Protocol for Scalable Asynchronous Multicast in P2P Systems," *Proc. IEEE INFOCOM,* Mar. 2005.

[21] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks," *IEEE J. Selected Areas in Comm.,* vol. 22, no. 1, pp. 91-106, Jan. 2004.

[22] T.T. Do, K.A. Hua, and M.A. Tantaoui, "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment," *Proc. IEEE Int'l Conf. Comm. (ICC),* June 2004.

[23] C. Gkantsidis and P.R. Rodriguez, "Network Coding for Large Scale Content Distribution," *Proc. IEEE INFOCOM '05,* Mar. 2005.

[24] M. Kim, L. Lima, F. Zhao, J. Barros, M. Medard, R. Koetter, T. Kalker, and K.J. Han, "On Counteracting Byzantine Attacks in Network Coded Peer-to-Peer Networks," *IEEE J. Selected Areas in Comm.,* vol. 28, no. 5, pp. 692-702, May 2010.

[25] F.H.P. Fitzek, M. Katz, and Q. Zhang, "Cellular Controlled Short-Range Communication for Cooperative P2P Networking," *Wireless Personal Comm.,* vol. 48, no. 1, pp. 141-155, 2009.

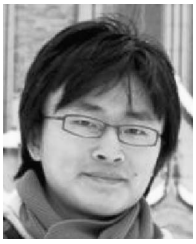
[26] F.H.P. Fitzek, M.V. Pedersen, J. Heide, and M. Médard, "Network Coding: Applications and Implementations on Mobile Devices," *Proc. Fifth ACM Workshop Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N),* pp. 83-87, Oct. 2010.

[27] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. Information Theory,* vol. 56, no. 9, pp. 4539-4551, Sept. 2010.

[28] F. Liu, S. Shen, B. Li, B. Li, H. Yin, and S. Li, "Novasky: Cinematic-Quality VoD in a P2P Storage Cloud," *Proc. IEEE INFOCOM,* pp. 936-944, Apr. 2011.

Yung-Cheng Kao received the MS and PhD degrees in computer science and engineering from Sun Yat-Sen University, Kaohsiung, Taiwan, in 2003 and 2010, respectively. His research interests include multimedia distribution network, wireless network, and peer-to-peer streaming network. He is currently an engineer of Industrial Technology Research Institute.

Chung-Nan Lee received the BS and MS degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, in 1980 and 1982, respectively, and the PhD degree in electrical engineering from the University of Washington, Seattle, WA, in 1992. Since 1992, he has been with National Sun Yat-Sen University, Kaohsiung, Taiwan, where he was an associate professor in the Department of Computer Science and Engineering from 1992 to 1999; he was a chairman of the Department of Computer Science and Engineering from August 1999 to July 2001 and is currently a professor and director of Cloud Computing Research Center. His current research interests include multimedia over wireless networks, cloud computing, and evolutionary computing.



Peng-Jung Wu received the BS degree in computer information science from Soochow University, Taipei, Taiwan, in 2001, and the MS degree in computer science and engineering from Sun Yat-Sen University, Kaohsiung, Taiwan, in 2003. He is working toward the PhD degree at the Department of Computer Science and Engineering in Sun Yat-Sen University, Kaohsiung, Taiwan. He was also a visiting scholar at the Department of Electrical Engineering, University of Washington, Seattle, from 2008 to 2009. His research interests include wireless network and peer-to-peer streaming system. He is currently an engineer of Industrial Technology Research Institute.

Hui-Hsiang Kao received the MS degree in computer science and engineering from National Sun Yat-sen University, Kaohisung, Taiwan, in 2006. He is currently working toward the PhD degree in the Department of Computer Science and Engineering at the National Sun Yat-Sen University. His research interests include wireless network and computer networks.